JRC Scientific and Technical Reports

Services State of Play

Compliance Testing and Interoperability Checking

S. Viganó
TXT e-solutions

M. Millot
European Commission DG JRC,
Institute for Environment and Sustainability

EUR 23049 EN - 2007

JRC EUROPEAN COMMISSION

ies Institute for Environment and Sustainability

TXT e-solutions

The mission of the Institute for Environment and Sustainability is to provide scientific-technical support to the European Union's Policies for the protection and sustainable development of the European and global environment.

European Commission
Joint Research Centre
Institute for Environment and Sustainability

Contact information

Michel Millot

Address:
European Commission, Joint Research Centre,
Institute for Environment and Sustainability
Spatial Data Infrastructure Unit
T.P. 262
Via E. Fermi, 2749
I-21027 Ispra (VA)
ITALY

E-mail: michel.millot@jrc.it
Tel.: +39-0332786146
Fax: +39-0332786325

http://ies.jrc.ec.europa.eu
http://www.jrc.ec.europa.eu

Stefano Viganó

Address:
TXT e-solutions
Via Frigia, 27
I-20126 Milano (MI)
ITALY

E-Mail: stefano.vigano@txt.it
Tel. +39-02257711
Fax.+39-022578994

http://www.txtgroup.com/it/index.shtml

Legal Notice

JRC 41755

EUR 23049 EN
ISBN 978-92-79-07820-0
ISSN 1018-5593
DOI 10.2788/55814

Luxembourg: Office for Official Publications of the European Communities

Printed in Italy

Table of Contents

1. Introduction

The European Community has adopted a directive of the European Parliament concerning the establishing of an Infrastructure for Spatial Information in Europe (INSPIRE)[1].

The overall aim of the INSPIRE proposal is to improve the way in which spatial data held by public authorities supports environmental policy, by improving the harmonisation of spatial data and the interoperability of spatial services and ensuring greater sharing of the data between public authorities and on-line access by the public.

In the month of April 2007 this directive has entered into force (transposition phase) and now the European public authorities have a period of two years to bring into force laws, regulations and administrative provisions necessary to comply to the INSPIRE directive.

In detail, they will have to adopt the Implementing Rules for the:

- creation and up-dating of the metadata
- network services
- third parties use of the upload services
- monitoring and reporting
- governing access and rights of use to spatial data sets and services for Community institutions and bodies

This document will focus on the network services issue (in the list above), trying to analyse the present situation in terms of normative (with respect to INSPIRE requirements), market solutions and test cases. Consequently, the starting point of this survey must necessarily be INSPIRE itself.

The work programme of the INSPIRE transposition phase (excerpt from [1]), regarding the Network Services and Interoperability, asserts that the member states shall operate a network of the following services available to the public for data sets and services for which metadata has been created:

- Discovery services; No charge
- View services; No charge (with exceptions)
- Download services;
- Transformation services,
- Services allowing spatial data services to be invoked;

An Implementing Rule shall be adopted for the different types of service according to the INSPIRE Roadmap while a different Implementing Rule shall be adopted for interoperability and where practical for harmonisation of spatial data sets and services.

These Implementing Rules will be adopted within two years after entry into force for data sets corresponding to the data themes described in the Annex I of [1] to the INSPIRE Directive and within 5 years for those covered in Annex II and III of the same document.

The directive requires that network services must be accessible through an EU geo-portal; therefore the geo-portal software architecture shall enforce the inter-operability with the Member States network. In particular the following issues will be addressed:

- General architectural model
- Security (access to the service and data transfer) when applicable
- Multilingualism

[1] http://www.ec-gis.org/inspire/

- Metadata for services
- Compliancy with services metadata and impact
- Technical architectures and protocols
- End-users' needs.

1.1 Network Services as Web Services

This document will assume that the general definition of a network service adopted by the INSPIRE directive can be regarded as a (more accurate) Web service. As a matter of fact, the final target of EC directive will be the implementation of a geo-portal software architecture ensuring the inter-operability of different actors through Internet.
We can therefore say, along with the W3C organisation[2], that *"a Web service (many sources also capitalize the second word, as in Web Services) is a software system designed to support interoperable Machine to Machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services"*.

The W3C Web service definition encompasses many different systems, but in a common usage the term refers to clients and servers that communicate using XML messages that follow the SOAP standard. Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server, a description in the Web Services Description Language (WSDL). The latter is not a requirement of a SOAP endpoint, but it is a prerequisite for automated client-side code generation in the mainstream Java and .NET SOAP frameworks. Some industry organizations, such as the WS-I, mandate both SOAP and WSDL in their definition of a Web service.

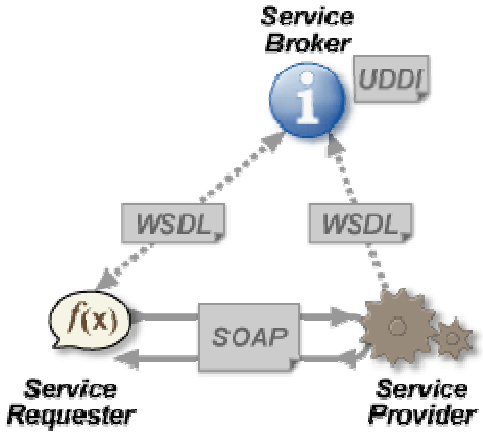

Figure 1.1 - Schematic Representation of a Web Service (source: Wikipedia)

1.2 Core WS specifications

The specifications that define Web services are intentionally modular and, as a result, there is no one document that contains them all. Additionally, there is neither a single, nor a stable set of specifications. There are a few "core" specifications that are supplemented by others as the circumstances and choices of technology dictate, including:

- **SOAP**

[2] Quoted from Wikipedia definition of Web Service

It's an XML-based, extensible message envelope format, with "bindings" to underlying protocols. The primary protocols are HTTP[3] and HTTPS, although bindings for others, including SMTP and XMPP, have been written.
According to W3C, *"SOAP is a lightweight protocol for exchanging structured information in a decentralized, distributed environment and it consists of three parts:*

- *an envelope that defines a framework for describing what is in a message and how to process it;*
- *a set of encoding rules for expressing instances of application-defined data types;*
- *a convention for representing remote procedure calls and responses";*

- **WSDL**
  It's an XML format that allows service interfaces to be described, along with the details of their bindings to specific protocols. Typically used to generate server and client code, and for configuration.
- **UDDI**
  It's a protocol for publishing and discovering metadata about Web services, to enable applications to find Web services, either at design time or runtime.
  Most of these core specifications have come from W3C, including XML, SOAP and WSDL; UDDI comes from OASIS.

The technologies named above have been listed according to the service stack, from the lowest transport level (SOAP) to the highest level (UDDI).

[3] The HTTP/1.1 specification is published by the Internet Engineering Task Force (IETF) as RFC 2616

The typical stack of a web service implementation can be described in the following schema:

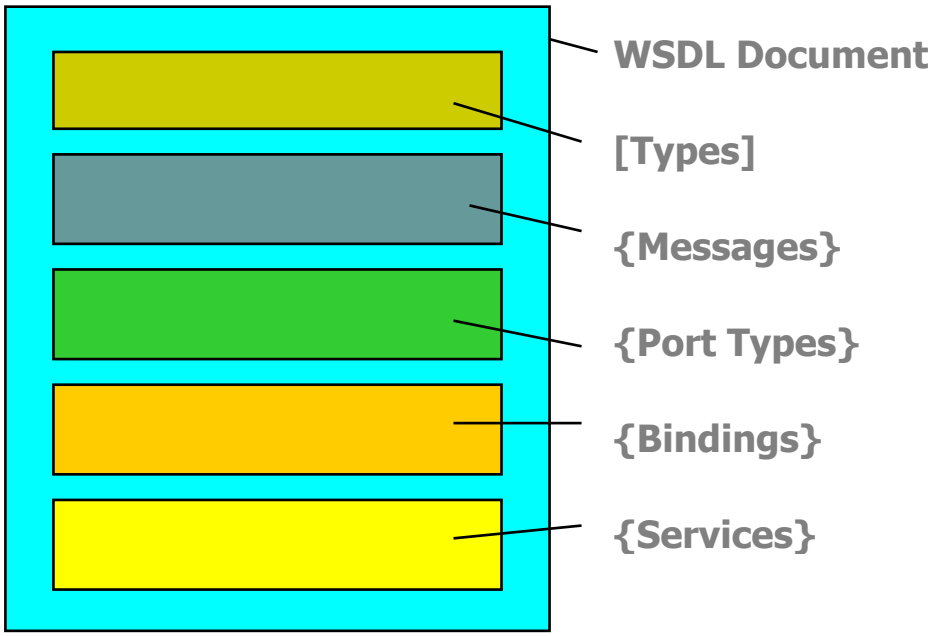

Figure 1.2 - Web Service Stack[4]

1.3 Profiles

To improve interoperability of Web Services, the WS-I publishes profiles. A profile is a set of core specifications (SOAP, WSDL ...) in a specific version (SOAP 1.1, UDDI 2 ...) with some additional requirements to restrict the use of the core specifications. The WS-I also publishes use cases and test tools to help deploying profile compliant Web services.

A profile identifies the use of particular options available in one or more base standards and it provides a basis for developing conformance tests. Furthermore, a compliant profile must not contradict the base specifications or otherwise give rise to nonconforming conditions.

1.4 Current situation

Many standards are available today. The list below[5], far from be exhaustive, is just a sample of the existing situation:

- Business Process Specifications
  - WS-BPEL
  - WS-CDL[6]
- Directory Access
  - Universal Description, Discovery and Integration (UDDI 1.0, 2.0 and 3.0)
  - ebXML
  - WSFL
  - WS-* (-Discovery, -Policy*, etc.)

[4] From the Powerpoint reference [12]
[5] http://en.wikipedia.org/wiki/List_of_Web_service_specifications
[6] XML-based language that describes peer-to-peer collaborations of Web Services participants

- Service Description (Metadata)
  - Web Services Description Language (WSDL) from the W3C
  - Web Services Semantics (WSDL-S)
  - XINS provides a POX-style Web service specification format
  - WS-MetadataExchange
  - WS-Resource Framework (WSRF)
- Messaging and Function Calls
  - Simple Object Access Protocol (SOAP)
  - SOAP variants (with Attachments[7], over-UDP, MTOM, etc.)
  - XML-RPC (XML-based Remote Procedure Call)
  - MTOM
  - XOP (XML-binary Optimized Packaging)
  - WS-* (like –Eventing, -Addressing, Transfer etc.)

The Figure 1.3 below is a sample schema of the different layers outlined in the previous list.

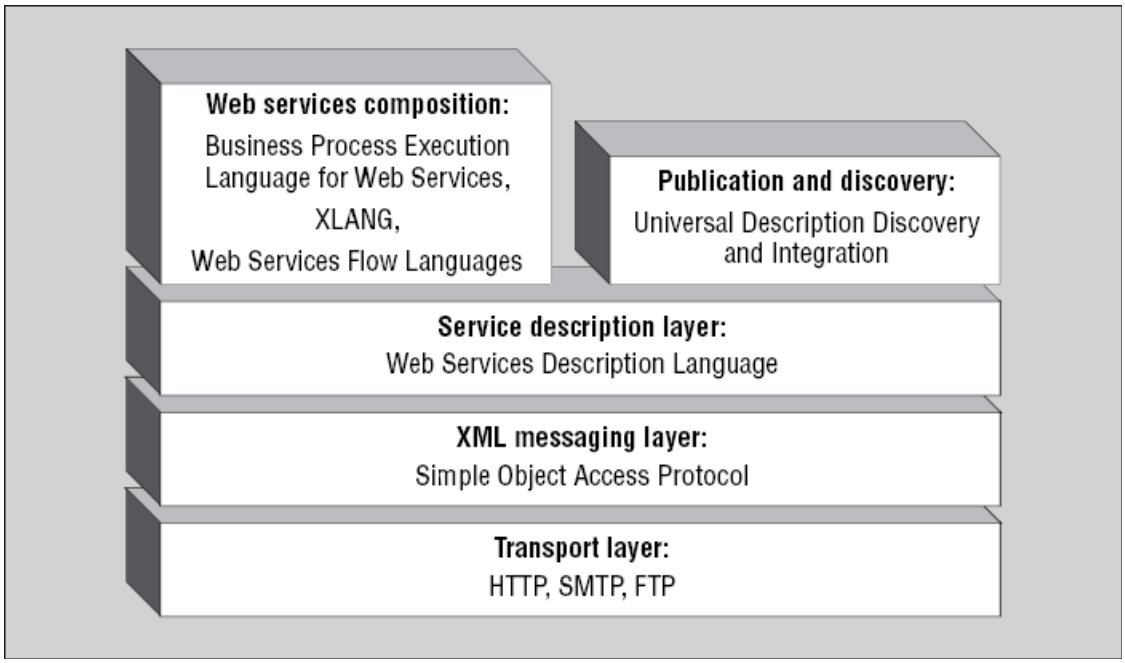

Figure 1.3 - Publication, Discovery and Interoperability Layers[8]

[7] It combines SOAP and MIME over HTTP.
[8] From the web article available at http://hcs.science.uva.nl/semanticweb/literatuur/x1072.pdf

2. Purpose and Scope of the Document

The first chapter of this document introduces the object of this survey, i.e. the Web services, starting the inquiry with the European Community INSPIRE directive [1].

The chapter gives a general description of what a Web service is up to this day and the different standards and technologies that software teams can adopt to develop Web Services.

The third chapter talks about the existing normative background related to standards for web service. The discussion is not limited to the INSPIRE directive but it tries to evaluate the difference with the most common standards.

In chapter 4 you can find a description of the INSPIRE requirements for service interoperability and compliancy, trying to evaluate these requirements in the light of existing standards.

In chapter 5 the document thereafter focuses the subject on the conformance of some market solutions: a few software packages and solutions are taken into account with an analytic approach.

The chapter 6 tries to review the service state of play under a critical point of view, underlying what is present and what is missing.

In the next chapter the author describes a few test cases related to some of the software described in chapter 5 and in the last chapter there is the conclusion of the survey.

The document is then closed with the required annexes for the used terminology and reference bibliography.

3. Normative References

The Inspire programme has released the document [1] in order to give to Drafting Teams guidance for the definition of Network Services Implementing Rules.

According to this document, the Network Services IR shall be based on the following principles:

- Ease of use and accessibility via the Internet or any other appropriate means of telecommunication available to the public (Article 18, 1).
- Take into account technological progress and minimum performance criteria (Article 22, a)
- Based on infrastructures for spatial information established and operated by the Member States (Directive recitals).

In detail, the following sentence has been quoted from the Article 18 of the directive requirements:

*"Member States shall establish and operate a **network** of the following **services** for the spatial data sets and services for which metadata have been created in accordance with this Directive:*

a) discovery services making it possible to search for spatial data sets and spatial data services on the basis of the content of the corresponding metadata and to display the content of the metadata;

b) view services making it possible, as a minimum, to display, navigate, zoom in/out, pan, or overlay spatial data sets and to display legend information and any relevant content of metadata;

c) download services, enabling copies of complete spatial data sets, or of parts of such sets, to be downloaded;

d) transformation services, enabling spatial data sets to be transformed;

e) "invoke spatial data services" services, enabling data services to be invoked.

Those services shall be easy to use and accessible via the Internet or any other appropriate means of telecommunication available to the public".

In detail, the Implementing Rules for the Network Services will address:

- General architectural model
- Security (access to the service and data transfer) when applicable
- Multilingualism as requested by INSPIRE.
- Compliance with services metadata and impact
- Technical architectures and protocols
- End-users' needs.

The INSPIRE directive states that a network service is what the W3C has defined as *Web Services*[9]. According to this definition, a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

The following requirements (see previous quotation) have been expressed on the functional side:

- Upload Services (to make metadata, data and services accessible)

[9] http://www.w3c.org/TR/ws-arch/

- Discovery Services (allowing to look for and data and services)
- View Services (related to the front-end navigation in the spatial data sets)
- Download Services (both partial and full copy of spatial datasets)

These requirements will be detailed in Chapter 4.

The most important issue among the previous ones is related to the discovery services giving, as a minimum, the following combination of search criteria to be implemented:

a) keywords;
b) classification of spatial data and services;
c) spatial data quality and accuracy;
d) degree of conformity with the harmonised specifications provided for in Article 11;
e) geographical location;
f) conditions applying to the access to and use of spatial data sets and services;
g) the public authorities responsible for the establishment, management, maintenance and distribution of spatial data sets and services.

Different standards are spreading through the web and the present scenario offers manifold options to organisations addressed by the INSPIRE directive.

The most relevant source for standards, definitions and general documentation references related to the web services are the W3C and OASIS organisations. Both have proposed different standards to face the challenge of more complex and more conceptual-level interfaces between web applications. Other organisations anyway, like ISO and OGC, have also proposed their standards.

The current situation (yet still evolving) is outlined in the following table:

Organisation	**Web Services standards**
ISO	TC211 – ISO 19119:2005…
W3C	WSDL
OASIS	BPEL4WS, ebXML
OGC	CSW

Table 3.1: Standards and organisations

In the web sites pages of the listed organisation it is possible to look and find technical reports, recommendations and working draft documents of interest on the web services subject. Finally, there is also the *Barry & Associates* web site[10] with overall and comprehensive information on web services.

The next paragraphs will try to make an overview of the standards for services, proposed by these organisations.

[10] http://www.service-architecture.com/web-services/articles/index.html

3.1 ISO 19119:2005

The technical committee TC211 of the ISO organisation has published a set of standards (IS) related to geomatic and geographic information. The most interesting one worth to mention in the scope of this document is the ISO 19119 (along with ISO 19115).

This standard should provide a framework for developers to create software that enables users to access and process geographic data from a variety of sources across a generic computing interface within an open information technology environment.

The geographic services architecture specified in ISO 19119 has been developed[11] to meet the following purposes:

- provide an abstract framework to allow coordinated development of specific services
- enable interoperable services through interface standardization
- support development of a service catalogue through the definition of service metadata
- allow separation of data instances & service instances
- enable use of one provider's service on another provider's data
- define an abstract framework which can be implemented in multiple ways

ISO 19119 is based on the Reference Model of Open Distributed Processing [ISO/IEC 10746].

Several terms are used extensively in the ISO 19119 definition:

- Service: Distinct part of the functionality that is provided by an entity through interfaces.
- Interface: Named set of operations that characterize the behaviour of an entity.
- Operation: Specification of a transformation or query that an object may be called to execute. It has a name and a list of parameters.

Services are accessible through a set of interfaces that are a set of operations.

The aggregation of interfaces in a service defines functionality of value to the users, i.e. software agents or human users. A service provides functionality that adds value. The value is apparent to the user who invoked the service.

The aggregation of operations in an interface and the definition of interface are for the purpose of software reusability. Interfaces are defined in order to be reusable for multiple service types. The syntax of an interface may be reused with multiple services with different semantics.

Interfaces are defined through operations. An operation specifies a transformation on the state of the target object or a query that returns a value to the caller of the operation.

To evaluate the fitness for use of a service in a specific context, users will review a description of the service. These service descriptions are also called *service metadata*.
Service metadata records can be managed and searched using a catalogue service as is done for dataset metadata. In order to provide a catalogue for discovering services, a schema for describing a service is needed.

The Catalogue Service Implementation Specification 2.0 (CSW 2.0) constitutes a general model for catalogue services. CSW 2.0 uses the concept of profiles as a basis for the development of application profiles for Catalogue Services. An application profile specifies the use of an application-layer protocol.

The developed Catalogue Application Profile defines an information model based on ISO 19115/ISO19119 and specifies a HTTP/1.1 protocol binding with support for SOAP messaging.

[11] Developed jointly with the Services Architecture SIG of the OpenGIS Consortium (OGC)

By applying the concept of profiles it is possible to achieve interoperability between different Catalogue Service implementations: each implementation must comply with the base specification given by CSW 2.0 and can hence interoperate on that level.

3.2 W3C - WSDL

The W3C Consortium has recently published the final technical document of WSDL 2.0 and it has made it a Recommendation, i.e. a standard.

WSDL 2.0 inherits all the specifications introduced with WS-I Basic Profile (see document [3]) and better addresses or defines services characteristics like inheritance, functionality import, error description and the "full" support to SOAP and HTTP.

You can find a complete and exhaustive documentation on WSDL 2.0 at the W3C dedicated pages[12]. The next paragraph will be dedicated to peruse the WSDL 2.0 specifications.

Quoting from [10], *"WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST and MIME"*.

A WSDL document defines services as collections of network endpoints called ports. The abstract definition of a single endpoint and the related messages is separated from their concrete network deployment or data format bindings, allowing for reusability of these (abstract) definitions.

A message is the abstract descriptions of the data being exchanged while the port types are the abstract collections of operations. Given a port type, protocol and data format specification constitutes a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service.

Concerning the type definition language, WSDL supports the XML Schemas specification (XSD) as its canonical type system, allowing anyway the usage of other type definition languages via extensibility.

In addition, WSDL defines a common binding mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

The WSDL specification also exploits the binding extensions for the following protocols and message formats:

- SOAP 1.1 (see Section 3 of document [3])
- HTTP GET / POST (see Section 4)
- MIME (see Section 5)

[12] http://www.w3.org/TR/wsdl20

3.3 Service Oriented Architecture

A new approach to define and implement services over the network (with a wide meaning) is the recent Service Oriented Architecture model.

SOA is a design for linking business and computational resources (principally organizations, applications and data) on demand to achieve the desired results for service consumers (which can be end users or other services). The following definition, given in [6], is now common for the Web services and the related architectures.

"A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed".

The basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services. SOA Web services are supported by most major software vendors and industry analysts. Unlike RPC Web services, loose coupling is more likely, because the focus is on the "contract" that WSDL provides, rather than the underlying implementation details.

Hence, the advantages of a SOA are loose coupling and asynchronous communication, as well as ease of maintenance, configuration and diagnosis. Standard protocols (Web service and SOAP family of standards) facilitate integration, interoperability and increase vendor independence. Distributed Web services are naturally fit for message-type communication in scalable architectures (for high throughput, data-intensive communication, low overhead protocols are often chosen).

Service-oriented architectures can be considered the evolution of existing (in the past) DCOM or Object Request Brokers (ORBs) based on the CORBA specification.

SOAs using Web services is considered as the state-of-the-art approach to support interoperability between distributed systems and therefore facilitates complex interactions between heterogeneous and autonomous systems both within the enterprise and for cross-organizational collaboration.

Message-based interactions are seen as the core building block in this new document-centric computing paradigm. Building SOA-based applications is a complex undertaking, design patterns and frameworks have an important role to play to ease the process and to provide high interoperability.

3.3.1 SOA Components

SOA architectures consist of the following three components:

- Service provider
- Service consumer
- Service registry

Each component can also act as one of the two other components. For instance, if a service provider needs additional information that it can only acquire from another service, it acts as a service consumer. Refer to Figure 3.1 for the operations that each component can perform.

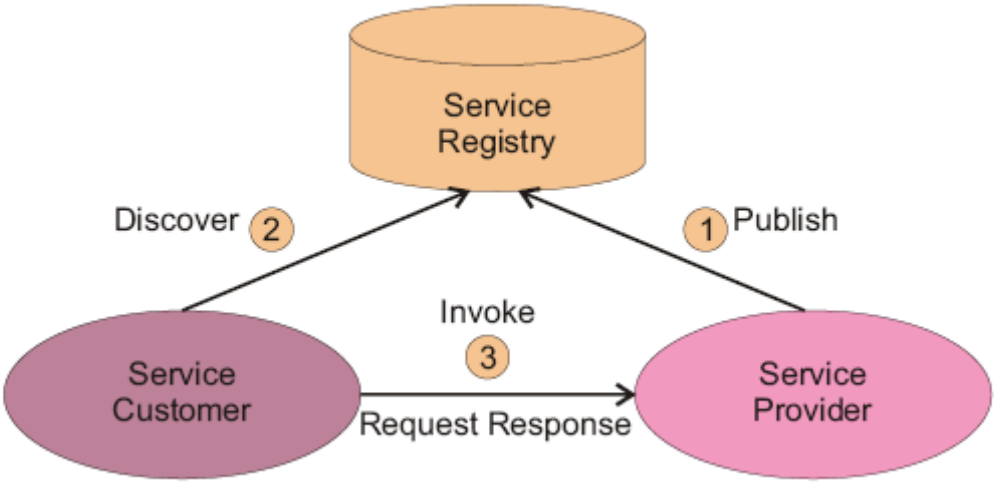

Figure 3.1 - SOA Components

The service provider creates a service and in some cases publishes its interface and access information to a service registry. The service registry is responsible for making the service interface and implementation access information available to service consumers. The service consumer locates entries in the service registry and then binds to the service provider in order to invoke the defined service.

3.3.2 SOA Stack

SOA may be built on Web services standards (e.g., using SOAP) that have gained broad industry acceptance. These standards (also referred to as web service specifications) also provide greater interoperability and some protection from lock-in to proprietary vendor software. One can, however, implement SOA using any service-based technology, such as Jini.

Service-oriented architecture is often defined as services exposed using the Web Services Protocol Stack. The base level of web services standards relevant to SOA includes the following:

- XML - a markup language for describing data in message payloads in a document format
- HTTP (or HTTPS) - request/response protocol between clients and servers used to transfer or convey information
- SOAP - a protocol for exchanging XML-based messages over a computer network, normally using HTTP
- XACML - a markup language for expressing access control rules and policies.
- Web Services Description Language (WSDL) - XML-based service description that describes the public interface, protocol bindings and message formats required to interact with a web service
- Universal Description, Discovery, and Integration (UDDI) - An XML-based registry to publish service descriptions (WSDL) and allow their discovery

All these standards, however, are more relevant to SOA architectures, since a system does not necessarily need to use any or all of these standards to be "service-oriented." For example, some service oriented systems have been implemented using Corba, Jini and REST.

3.3.3 SOA Governance

SOA Governance is about ensuring that each new and existing service conforms to the standards, policies and objectives of an organization for the entire life of that service.

SOA Governance plays an increasingly important role in today's challenging business environment. It provides structure, commitment and support for the development, implementation and management of SOA, as necessary, to ensure it achieves its objectives.

3.3.4 SOA Testing

In SOA architectures, services are based on heterogeneous technologies. No longer can we expect to test an application that was developed by a unified group, as a single project, sitting on a single application server and delivering through a standardized browser interface. The ability to string together multiple types of components to form a business process requires unconstrained thinking from an architect's perspective, and test planning and scheduling complexities from a tester's perspective.

So, if our area of interest is the Web Services, we would not like to test the entire technology stack that makes up the application. Due to complexity of the architecture, testing SOA could be viewed as a complex computing problem. In this light, what is the best way to test SOA architecture?

The best practice consists of breaking down the architecture to its component parts, working from the most primitive to the most sophisticated, testing each component, then the integration of the holistic architecture. In other words, the architecture should be divided into domains, such as services, security, and governance and test each domain separately using the recommended approach and tools.

SOA is loosely coupled with complex interdependencies and a SOA testing approach must follow the same pattern.

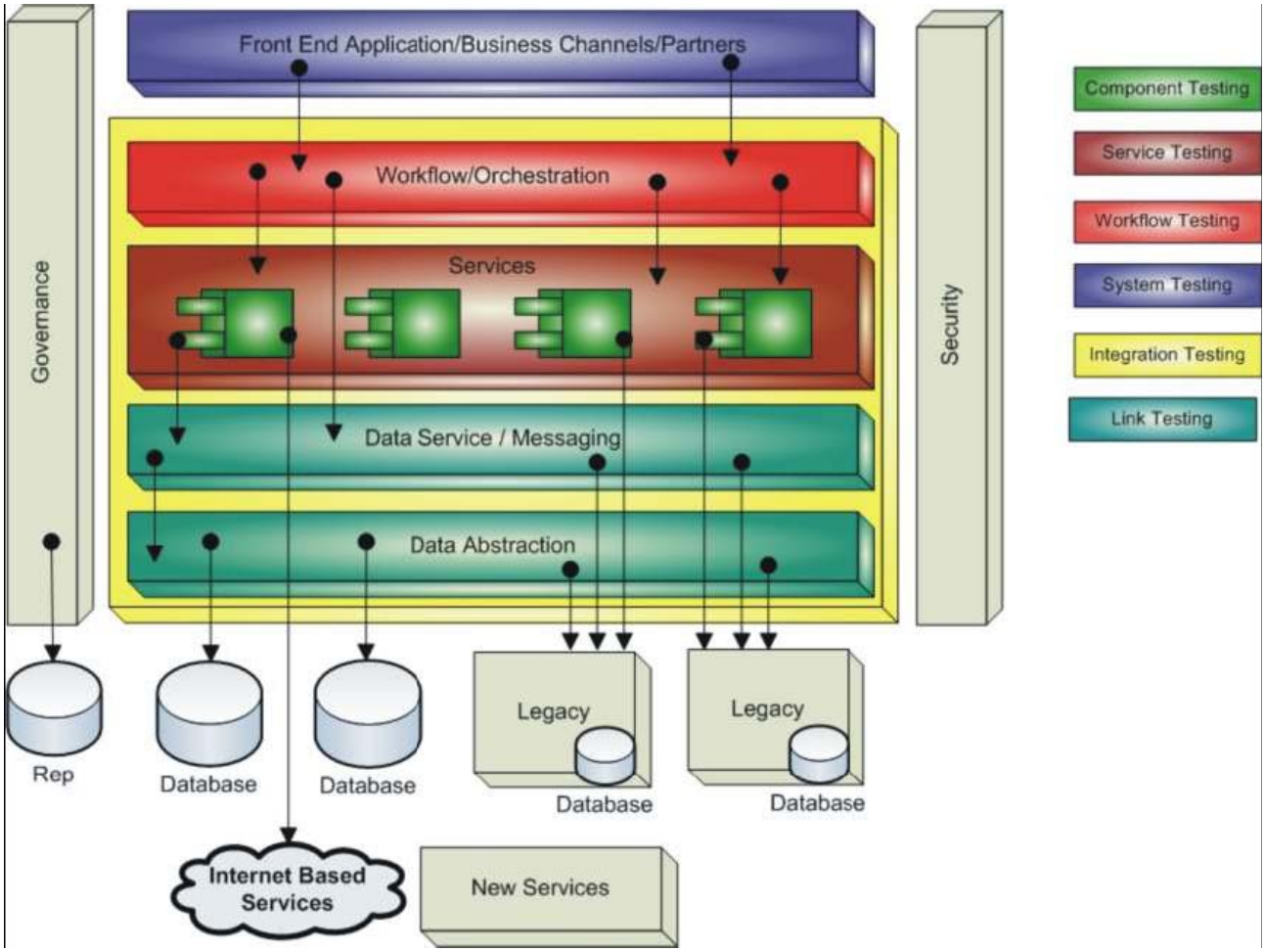

Figure 3.2 – SOA Levels and Components

Service testing will be the most important test level/phase within the SOA Test approach. Today, many organizations build a program or Web service, perform limited unit testing and accelerate its delivery to the integration test phase, to allow the test team to evaluate its quality. Service reuse will demand each service is delivered from this level/phase of testing with a comprehensive statement of quality and even a guarantee.

Service Level testing must ensure that the service is not only meeting the requirements of the current project, but more importantly, is still meeting the business and operational requirements of the other processes that are using that service.

As important in this architecture as the Service Level test, the Integration test focuses on service interfaces. This test phase aims to determine if interface behaviour and information sharing between the services, are working as specified. The test team must ensure that all the services delivered to this test phase comply with the defined interface definition, in terms of standards, format and data validation. Integration testing test scenarios should also 'work' the layers of communications, the network protocols. This test phase may include testing external services to the organization.

Process/Orchestration testing ensures services are operating collectively as specified. This phase of testing would cover business logic, sequencing, exception handling and process decomposition (including service and process reuse).

More test levels are required in the QA lifecycle of SOA solutions, like for example the Security or the System testing, but they are out of scope for the present study.

Sample technologies adopting SOA

Apache Axis, XFire, .NET and numerous other industry-specific products now have SOA built-in.

Standards for SOA

The following organisations support the SOA architecture: W3C, OASIS, IBM, BEA, and Microsoft.

3.4 BPEL for Web Services

According to reference [7], the goal of the Web Services effort is to achieve universal interoperability between applications by using Web standards. Web Services use a loosely coupled integration model to allow flexible integration of heterogeneous systems in a variety of domains including business-to-consumer, business-to-business and enterprise application integration.

Systems integration requires more than the ability to conduct simple interactions by using standard protocols. The full potential of Web Services as an integration platform can be achieved only when applications and business processes are able to integrate their complex interactions by using a standard process integration model.

The interaction model that is directly supported by WSDL is essentially a stateless model of synchronous or uncorrelated asynchronous interactions. Models for business interactions typically assume sequences of peer-to-peer message exchanges, both synchronous and asynchronous, within stateful, long-running interactions involving two or more parties.

To define such business interactions, a formal description of the message exchange protocols used by business processes in their interactions is needed. The definition of such business protocols involves precisely specifying the mutually visible message exchange behaviour of each of the parties involved in the protocol, without revealing their internal implementation. There are two good reasons to separate the public aspects of business process behaviour from internal or private aspects. One is that businesses obviously do not want to reveal all their internal decision making and data management to their business partners. The other is that, even where this is not the case, separating public from private process provides the freedom to change private aspects of the process implementation without affecting the public business protocol.

Business Process Execution Language (BPEL) for Web Services provides a means to formally specify business processes and interaction protocols. It represents a convergence of the ideas in the XLANG and WSFL specifications. Both XLANG and WSFL are superseded by the BPEL4WS specification. This language is supported by IBM, BEA Systems, Microsoft, SAP AG, Siebel Systems and a comprehensive documentation can be found in the IBM web site[13].

An accurate definition of BPEL4WS can be found in [7] that states: "*BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces*".

According to this definition, the main goal of BPEL4WS is orchestration, which is the composing of multiple web services into "loosely-coupled business flows". In the last years, the biggest problem occurred in asynchronous message-driven development has been the abundance of competing standards for orchestration. BPEL4WS is a step in the direction of making one standard, mixing block- and graph structured process models. This capability makes the language expressive and also very complex.

The basic concepts of BPEL4WS can be applied in one of two ways:

1. A BPEL4WS process can define a business protocol role, using the notion of abstract process. For example, in a supply-chain protocol, the buyer and the seller are two distinct roles, each with its own abstract process. Their relationship is typically modelled as a partner link. Abstract processes use all the concepts of BPEL4WS but approach data handling in a way that reflects the level of abstraction required to describe public aspects of the business protocol. Specifically, abstract processes handle only protocol-relevant data. BPEL4WS provides a way to identify protocol-relevant data as message properties. In addition, abstract processes use nondeterministic data values to hide private aspects of behaviour.
2. It is also possible to use BPEL4WS to define an executable business process. The logic and state of the process determine the nature and sequence of the Web Service interactions conducted at each business partner, and thus the interaction protocols. While a BPEL4WS process definition is not required to be complete from a private implementation point of view, the language effectively defines a portable execution format for business processes that rely exclusively on Web Service resources and XML data. Moreover, such processes execute and interact with their partners in a consistent way regardless of the supporting platform or programming model used by the implementation of the hosting environment.

BPEL4WS also uses XML Schema, SOAP and WSDL and the relation with these standards will be detailed in the next subparagraphs.

[13] http://www.ibm.com/developerworks/library/specification/ws-bpel/

3.4.1 Relationship with WSDL

BPEL4WS depends on the following XML-based specifications: WSDL 1.1, XML Schema 1.0, XPath 1.0 and WS-Addressing.

Among these, WSDL has the most influence on the BPEL4WS language. The BPEL4WS process model is layered on top of the service model defined by WSDL 1.1. At the core of the BPEL4WS process model is the notion of peer-to-peer interaction between services described in WSDL; both the process and its partners are modelled as WSDL services. A business process defines how to coordinate the interactions between a process instance and its partners. In this sense, a BPEL4WS process definition provides and/or uses one or more WSDL services, and provides the description of the behaviour and interactions of a process instance relative to its partners and resources through Web Service interfaces. That is, BPEL4WS defines the message exchange protocols followed by the business process of a specific role in the interaction.

The definition of a BPEL4WS business process also follows the WSDL model of separation between the abstract message contents used by the business process and deployment information (messages and portType versus binding and address information). In particular, a BPEL4WS process represents all partners and interactions with these partners in terms of abstract WSDL interfaces (portTypes and operations); no references are made to the actual services used by a process instance.

In the 1.1 version of the specification the core concepts are clearly separated from the extensions required specifically for the two usage patterns.

3.4.2 Attributes and defaults

The following list specifies the defaults for all standard attributes at the process and activity level. The table does not include activity-specific attributes (such as partnerLink in an invoke activity).

Parameter	**Default**
queryLanguage	http://www.w3.org/TR/1999/REC-xpath-19991116
expressionLanguage	http://www.w3.org/TR/1999/REC-xpath-19991116
suppressJoinFailure	no
variableAccessSerializable	no
abstractProcess	no
initiate	no
pattern	No default
createInstance	no
enableInstanceCompensation	no
joinCondition	Disjunction of the status of the incoming links
transitionCondition	true

Table 3.2 – Attributes and Defaults

BPEL is a substrate that binds pre-existing Web services together and is not intended to be used to implement the services themselves. This is one of the reasons that the feature set in BPEL is constrained to simple expressions and constructs. BPEL allows you to form new Web services through recombination and enables the assembly of complex business processes.

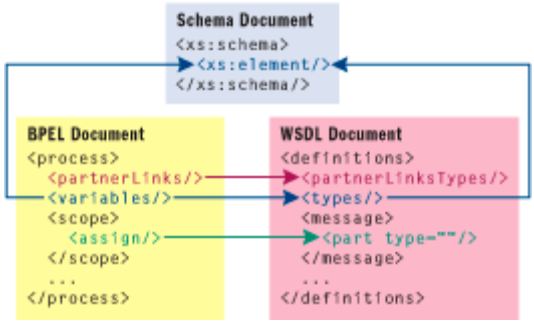

Figure 3.3 - Relationship between BPEL, WSDL, and XML Schema

A BPEL definition generally requires two other document types, WSDL and XML schema. BPEL extends WSDL to both provide and consume Web services in an abstract way. This enables one of the great strengths of WSDL, namely the ability to separate the abstract message and port information required at composition time from the physical binding and address details required at invocation time.

3.4.3 Conclusion

BPEL4WS is going to be a part of a standard based, internet scale backbone for application-to-application integration, but to fulfil this promise it needs some other components to interact with. Some of the other parts needed are already available and/or standardised – XML Schema, SOAP, WSDL and WS-Security. Other parts, WS-Addressing, WS-ReliableMessaging and WS-Transaction, are emerging to complete the needed components to make developers build the desired loosely coupled composite business flows.

There are other candidates for web service composition submitted to standardisation committees. Sun, BEA, SAP and Intalio are pushing for WSCI, Intalio also initiated BPMI.org with BPML and OASIS already supports BPSS which is a part of ebXML.

3.5 OASIS ebXML

Electronic Business using eXtensible Markup Language, commonly known as e-business XML, or ebXML as it is typically referred to as, is a family of XML based standards sponsored by OASIS and UN/CEFACT whose mission is to provide an open, XML-based infrastructure that enables the global use of electronic business information in an interoperable, secure, and consistent manner by all trading partners.

The ebXML work stemmed from earlier work on ooEDI (object oriented EDI), UML / UMM, XML markup technologies and the X12 EDI "Future Vision" work sponsored by ANSI X12 EDI. The melding of these components began in the original ebXML work and the theoretical discussion continues today. Other work relates, such as the Object Management Group work and the OASIS BCM (Business-Centric Methodology) standard (2006).

The International Standards Organization (IS0) has approved a suite of four ebXML OASIS Standards that enable enterprises in any industry, of any size, anywhere in the world to conduct business over the Internet. The submissions from OASIS have been published as ISO technical specifications, ISO/TS 15000.

The new ISO 15000 designation, under the general title, Electronic business eXtensible markup language, includes four parts, each corresponding to one of ebXML's modular suite of standards:

- ISO 15000-1: ebXML Collaborative Partner Profile Agreement
- ISO 15000-2: ebXML Messaging Service Specification
- ISO 15000-3: ebXML Registry Information Model
- ISO 15000-4: ebXML Registry Services Specification

EbXML provides companies with a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes. It aims to make it easier for organizations to interface with others within and outside their industry, up new markets with less effort than before and, at the same time, cut costs and simplify process associated with traditional document exchange.

What is unique about ebXML is that it's a complete standard, addressing:

- Process
- Trading partner management
- Semantics
- Notation
- Security
- Agreements
- Standard information exchange
- Standard information structure

Other standards, such as Business Process Execution Language for Web Services (BPEL4WS), address only the notion of process and semantics, or other more narrow aspects of application integration between trading partners or internal systems.

However, the aggressiveness of ebXML is also its most limiting factor, because it will take years before the standard finds its way into many enterprises and trading communities. This is due to the amount of work that must be done to get a trading community to leverage ebXML.

3.5.1 ebXML Components

There are several components to ebXML, including:

- Collaboration Protocol Profile (CPP)
- Collaboration Protocol Agreement (CPA)
- Business Process and Information Modelling
- Core components
- Messaging
- Registry/Repository

CPP describes an enterprise offering using a standard, portable format. This component describes the message-exchange mechanisms as well as business collaborations that are native to the enterprise or trading community. The ebXML standard also describes business processes within CPP, including how partners interact within a trading community. CPP supports intra- and inter-company processes, and public versus private processes, collaborating on both sides of a two-party B2B transaction. For example, when leveraging CPP, a trading community would define all processes between partners -for instance, buying parts to build a car -as well as semantic differences, and how processes and data need to interact to support any number of business activities.

CPA describes the particular requirements, facilities, and descriptions for the transaction of trading partner business. It is formed from either manual or automated systems, deriving the intersection of their agreed-upon CPPs. Thus, the CPA becomes the de facto contract between the trading partners, creating "rules of engagement" for a specific collaborative business transaction.

Business Process and Information Modelling is a specification for describing a business process in XML. This includes transactions, document flow, information encryption, binary collaborations, semantics, and such. Processes that leverage ebXML use these specifications when they create CPPs, which are also used to define shared business processes within a trading community.

3.5.2 OASIS Test Framework

On the OASIS web site you can find the document [5] that can help to understand the requirements of a test framework to state compliancy of a service solution with this standard. The Test Framework committee specifications can be found in [4], used also as the source for the following figures related to ebXML.

Quoting from the introduction paragraph of [5], "*…a test framework for automatically running test suites for - but not limited to - ebXML specifications. The framework includes an architecture design based on components that can be combined and distributed in different ways, to accommodate different test harnesses. It also includes an extensible test scripting language for coding test suites in an executable way. It can accommodate third-party plug-ins, that would perform advanced verifications for example on message material (e.g. semantic verification using rule engine), or that would help build testing material (e.g. digital signature).*"

The test framework is flexible enough to permit testing beyond ebXML message format. This framework describes six different phases (some mandatory, other not) to be implemented when testing for conformance or for interoperability:

1. Test Plan (not mandatory but recommended)
2. Test Requirements design
3. Test Harness design
4. Test Suite design

5. Validation Condition (not mandatory but recommended)
6. Test Execution

All these phases ensure conformance of the testing process to the OASIS specifications. All testing MUST follow the same procedural steps, and employ the same XML format.

The framework is built on the following hierarchy:

- Test Harnesses: they consists of different test configurations
- Test Driver: it interprets Test Case data and drives execution generating ebXML messages. It parse and interpret the Test Case definitions that are part of a Test Suite
- Test Service: it implements actions according to the incoming messages

The components of the framework are designed so that they can be combined in different configurations, or Test Harnesses.

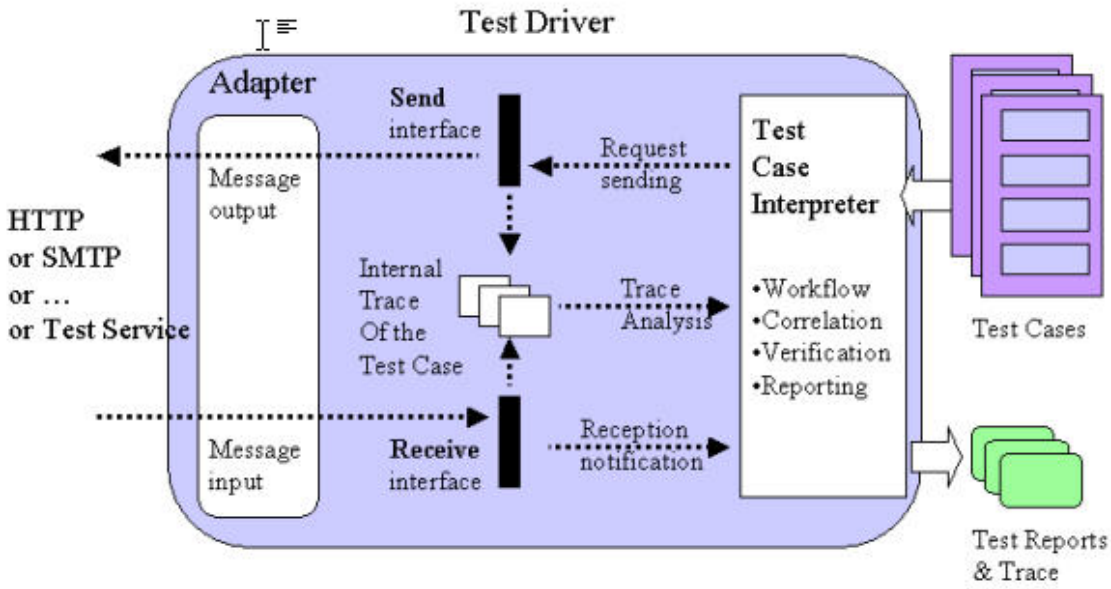

Figure 3.4 – Test Driver Schema

Users can define new Test Suites and Test Cases to be run on the framework. A Test Suite (either for conformance or for interoperability) can be run entirely and validated from one component of the framework, the Test Driver, (see Figure 3.4) which interprets Test Case data and drives Test Case execution.

The Test Driver can be used either in Connection Mode or in Service Mode.

When used in connection mode, the Test Driver is acting as a transport end-point that can receive or send messages with an envelope consistent with the transport protocol (e.g. HTTP, SMTP or FTP).

In service mode, the Test Driver directly interacts with the Service/Actions of the Test Service component, without involving the transport layer, e.g. by invoking these actions via a software interface, in the same process space.

3.5.3 The Test Service

The Test Service MAY NOT be a required component of the Test Framework. For conformance and interoperability testing of an ebXML Messaging Service implementation however, a Test Service is a REQUIRED Test Framework component. The Test Service represents the application layer for a message handler. Although the Test Service simulates an application, it is part of the Test Framework, and does not vary from one test harness to the other.

The Test Service operates in two modes: Reporting or Loop mode. In reporting mode the actions of the Test Service instance, when invoked, will send a notification to the Test Driver.

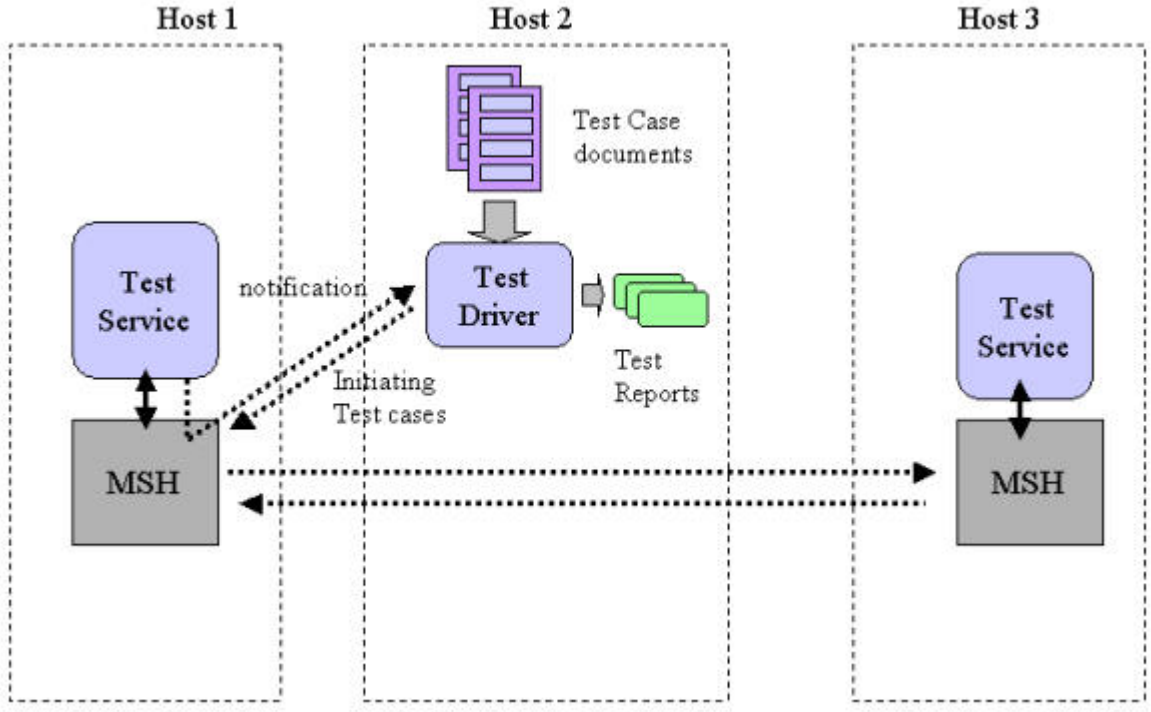

Figure 3.5 –Workflow for Interoperability

An example of the Test Service operating in remote mode for interoperability testing is showed in Figure 3.5.

OASIS Web Services Resource Framework (WSRF)

WSIF enables developers to interact with abstract representations of Web services through their WSDL descriptions instead of working directly with the Simple Object Access Protocol (SOAP) APIs, which is the usual programming model. With WSIF, developers can work with the same programming model regardless of how the Web service is implemented and accessed.

Apache CXF Service Framework

Apache CXF is an open source services framework and it's the evolution of the Xfire web services framework.

CXF helps you build and develop services using front-end programming APIs, like JAX-WS. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI.

CXF includes a broad feature set, but it is primarily focused on the following areas:

Web Services Standards Support: CXF supports a variety of web service standards including SOAP, the Basic Profile, WSDL, WS-Addressing, WS-Policy, WS-ReliableMessaging, and WS-Security.

Front-ends: CXF supports a variety of “front-end” programming models. CXF implements the JAX-WS APIs (version 2.0 will be TCK compliant). It also includes a “simple front-end” which allows creation of clients and endpoints without annotations. CXF supports both contract first development with WSDL and code first development starting from Java.

- **Ease of use**: CXF is designed to be intuitive and easy to use. There are simple APIs to quickly build code-first services, Maven plug-ins to make tooling integration easy, JAX-WS API support, Spring 2.0 XML support to make configuration a snap, and much more.
- **Binary and Legacy Protocol Support**: CXF has been designed to provide a pluggable architecture that supports not only XML but also non-XML type bindings, such as JSON and CORBA, in combination with any type of transport.

Services

If a service-oriented architecture is to be effective, we need a clear understanding of the term service. A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

Connections

The technology of Web services is the most likely connection technology of service-oriented architectures. Web services essentially use XML to create a robust connection.

4. Requirements Synthesis

The next paragraphs details the requirements expressed on the INSPIRE side for the conformance / interoperability of Network Services. This information has been obtained by [2].

4.1 High Level Requirements

The INSPIRE Proposal requires Member States:

- To establish and operate upload services for making metadata and spatial data sets and services accessible through the Network Services (Article 17).
- To establish and operate a network of the Network Services for the spatial data sets and services for which metadata have been created (Article 18).

The Network Services Implementing Rules shall be based on the following principles:

- ease of use and accessibility via the Internet or any other appropriate means of telecommunication available to the public (Article 18, 1);
- take into account technological progress and minimum performance criteria (Article 22, a);
- based on infrastructures for spatial information established and operated by the Member States (Directive recitals).

The Implementing Rules for the Network Services will in particular address:

- General architectural model
- Security (access to the service and data transfer) when applicable
- Multilingualism as requested by INSPIRE.
- Compliance with services metadata and impact
- Technical architectures and protocols
- End-users' needs.

And the development of the technical specifications of the Network Services, its schedule and priority setting will be driven by:

- Maturity of available reference materials

4.2 Detailed Requirements

According to [2], the INSPIRE functional requirements can be grouped in the following list of services, detailed in the following paragraphs.

- Upload services
- Discovery services
- View services
- Download services
- Transformation services
- Services allowing spatial data services to be invoked

4.2.1 Upload Services

Member States shall establish and operate upload services for making metadata and spatial data sets and services accessible through the services referred to in Article 18(1).

4.2.2 Discovery Services

The MS shall establish and operate discovery services making it possible to search for spatial data sets and spatial data services on the basis of the content of the corresponding metadata and to display the content of the metadata;

For the purposes of the discovery services, as a minimum the following combination of search criteria shall be implemented:

a) keywords;
b) classification of spatial data and services;
c) spatial data quality and accuracy;
d) degree of conformity with the harmonised specifications provided for in Article 11;
e) geographical location;
f) conditions applying to the access to and use of spatial data sets and services;
g) the public authorities responsible for the establishment, management, maintenance and distribution of spatial data sets and services.

4.2.3 View Services

The Member States shall establish and operated view services making it possible, as a minimum, to display, navigate, zoom in/out, pan, or overlay spatial data sets and to display legend information and any relevant content of metadata.

4.2.4 Download Services

The Member States shall establish and operate download services, enabling copies of complete spatial data sets, or of parts of such sets, to be downloaded. In addition, where public authorities levy charges for the download services, Member States shall ensure that e-commerce services are available.

4.2.5 Transformation Services

The Member States shall establish and operate transformation services, enabling spatial data sets to be transformed.

The transformation services shall be combined with the other services referred to in that paragraph in such a way as to enable all those services to be operated in conformity with the Implementing Rules laying down the following:

- harmonised spatial data specifications;
- Arrangements for the exchange of spatial data.

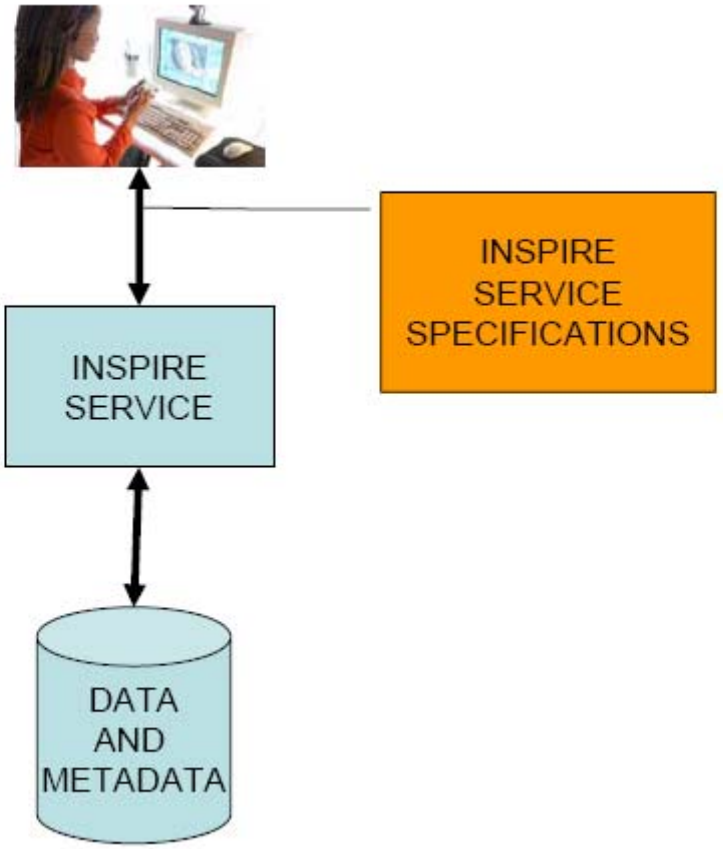

Figure 4.1 – INSPIRE Service Schema[14]

The previous specifications will be hereafter the object of IR Teams.

Different works have been done in the last years to define common standards for Web Services and SOA, trying to give to interfaces and communication protocols a set of requirements.

One of the results achieved by the OpenGIS Consortium has been the definition of a WCS specification that will work with the industry standards. This specification enables WCS to support WSDL and the SOAP protocol.

[14] The figure has been taken from reference [2]

Test requirements

In order to assert the conformance of a service to the Inspire directive

4.3 XML Web Services

XML-based Web Services can be described commonly by the following facts:

- They expose useful functionality to Web users through a standard Web protocol. In most cases, the protocol used is SOAP.
- They provide a way to describe their interfaces in enough detail to allow a user to build a client application to talk to them. This description is usually provided in an XML document called a Web Services Description Language (WSDL) document.
- They are registered so that potential users can find them easily. This is done with Universal Discovery Description and Integration (UDDI).

Even though all Web Services are different, consuming them always follows the exact same pattern. A web service must be implemented according to a given architecture (profile) where different standards attend to the general usability and behaviour of the service. The next figure displays an example of a common service stack.

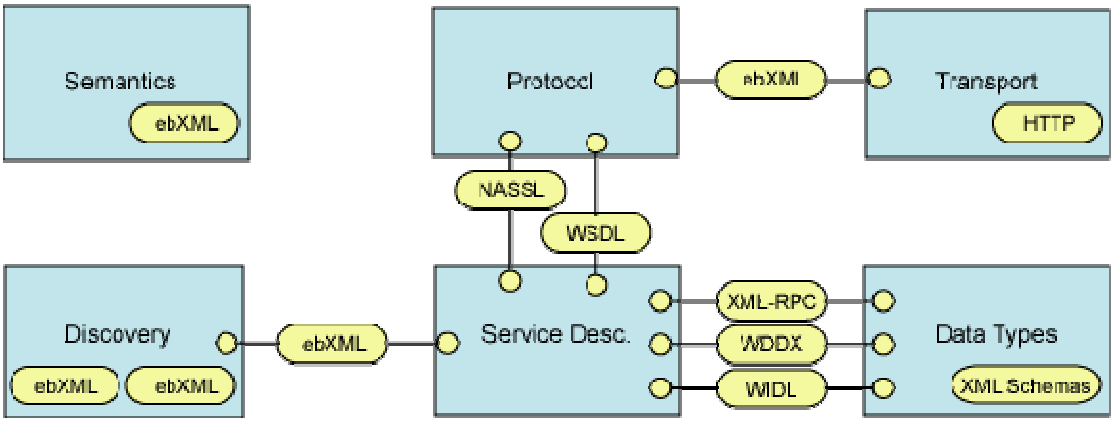

Figure 4.2 – Web Service architecture[15]

Once we decided to use a particular service, its formal description needs to be located. This is called the WSDL file, which is an XML document (not meant to be human readable) and usually published on the web via http transport layer.

WSDL provides an extensible mechanism for defining the base messaging description and metadata for a Web service, specifying what a request message must contain and what the response message will look like in unambiguous notation.

WSDL defines an XML-based grammar for describing network services as a set of endpoints that accept messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, which are bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

[15] The figure has been taken from http://www.ibm.com/developerworks/library/ws-soap/?dwzone=ws

WSDL is extensible to allow the description of endpoints and their messages regardless of what message formats or network protocols are being used to communicate.

In the next paragraphs I will take into account the required standards for the discovery and interoperability functionalities of a web service.

4.3.1 SOAP

SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework that more abstract layers can build on.

There are several different types of messaging patterns in SOAP, but by far the most common is the Remote Procedure Call (RPC) pattern, in which one network node (the client) sends a request message to another node (the server), and the server immediately sends a response message to the client. SOAP is the successor of XML-RPC, though it borrows its transport and interaction neutrality and the envelope/header/body from other protocols like WDDX.

Soap defines the XML format for messages, specifying exactly how to encode an HTTP header and an XML file so that a program in one computer can call a program in another computer and pass it information. It also specifies how the called program can return a response.

SOAP is similar in purpose to the DCOM and CORBA distributed object systems, but is lighter weight and less programming intensive. Because of its simple exchange mechanism, SOAP can also be used to implement a messaging system.

4.3.2 WSDL

WSDL (often pronounced whiz-dull) stands for Web Services Description Language. For our purposes we can say that a WSDL file is an XML document that describes a set of SOAP messages and how the messages are exchanged. In other words, WSDL is to SOAP what IDL is to CORBA or COM. Since WSDL is XML, it is readable and editable but in most cases, it is generated and consumed by software.

The notation that a WSDL file uses to describe message formats is based on the XML Schema standard which means it is both programming-language neutral and standards-based which makes it suitable for describing XML Web services interfaces that are accessible from a wide variety of platforms and programming languages. In addition to describing message contents, WSDL defines where the service is available and what communications protocol is used to talk to the service. This means that the WSDL file defines everything required to write a program to work with an XML Web service[16].

4.3.3 SOAP Binding

WSDL includes a binding for SOAP 1.1 endpoints, which supports the specification of the following protocol specific information:

- An indication that a binding is bound to the SOAP 1.1 protocol
- A way of specifying an address for a SOAP endpoint.
- The URI for the SOAPAction HTTP header for the HTTP binding of SOAP
- A list of definitions for Headers that are transmitted as part of the SOAP Envelope

[16] For detail refer to: http://www.w3.org/TR/wsdl

This binding grammar it is not an exhaustive specification since the set of SOAP bindings is evolving. Nothing precludes additional SOAP bindings to be derived from portions of this grammar. In the example below, a SubscribeToQuotes SOAP 1.1 one-way message is sent to a StockQuote service via a SMTP binding. The request takes a ticker symbol of type string, and includes a header defining the subscription URI.

```
<?xml version="1.0"?>
<definitions name="StockQuote"
          targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
    <message name="SubscribeToQuotes">
        <part name="body" element="xsd1:SubscribeToQuotes"/>
        <part name="subscribeheader" element="xsd1:SubscriptionHeader"/>
    </message>
    <portType name="StockQuotePortType">
        <operation name="SubscribeToQuotes">
           <input message="tns:SubscribeToQuotes"/>
        </operation>
    </portType>
    <binding name="StockQuoteSoap" type="tns:StockQuotePortType">
        <soap:binding style="document" transport="http://example.com/smtp"/>
        <operation name="SubscribeToQuotes">
           <input message="tns:SubscribeToQuotes">
               <soap:body parts="body" use="literal"/>
               <soap:header message="tns:SubscribeToQuotes" part="subscribeheader" use="literal"/>
           </input>
        </operation>
    </binding>
    <service name="StockQuoteService">
        <port name="StockQuotePort" binding="tns:StockQuoteSoap">
           <soap:address location="mailto:subscribe@example.com"/>
        </port>
    </service>
    <types>
        <schema targetNamespace="http://example.com/stockquote.xsd"
               xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="SubscribeToQuotes">
               <complexType>
                   <all>
                       <element name="tickerSymbol" type="string"/>
                   </all>
               </complexType>
           </element>
           <element name="SubscriptionHeader" type="uriReference"/>
        </schema>
    </types>
</definitions>
```

Listing 4.1 – SOAP binding of one-way operation over SMTP using a SOAP Header

In the example the XML SOAP extensions have been highlighted in blue. The SOAP binding generally extends WSDL with the extension elements displayed in blue colour in the listing below.

```
<definitions .... >
    <binding .... >
        <soap:binding style="rpc|document" transport="uri">
        <operation .... >
           <soap:operation soapAction="uri"? style="rpc|document"?>?
           <input>
               <soap:body parts="nmtokens"? use="literal|encoded"
                          encodingStyle="uri-list"? namespace="uri"?>
               <soap:header message="qname" part="nmtoken" use="literal|encoded"
                            encodingStyle="uri-list"? namespace="uri"?>*
                 <soap:headerfault message="qname" part="nmtoken" use="literal|encoded"
                                   encodingStyle="uri-list"? namespace="uri"?/>*
```

```
            <soap:header>
        </input>
        <output>
            <soap:body parts="nmtokens"? use="literal|encoded"
                       encodingStyle="uri-list"? namespace="uri"?>
            <soap:header message="qname" part="nmtoken" use="literal|encoded"
                        encodingStyle="uri-list"? namespace="uri"?>*
              <soap:headerfault message="qname" part="nmtoken" use="literal|encoded"
                                encodingStyle="uri-list"? namespace="uri"?/>*
            <soap:header>
        </output>
        <fault>*
            <soap:fault name="nmtoken" use="literal|encoded"
                       encodingStyle="uri-list"? namespace="uri"?>
         </fault>
      </operation>
  </binding>

  <port .... >
      <soap:address location="uri"/>
  </port>
</definitions>
```

Listing 4.2 – SOAP binding extensions for WSDL

4.3.4 UDDI

UDDI is a directory service where businesses can register and search for Web services. UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.

Key facts related to UDDI:

- UDDI stands for Universal Description, Discovery and Integration
- UDDI is a directory (yellow pages) for storing information about web services
- UDDI is a directory of web service interfaces described by WSDL
- UDDI communicates via SOAP
- UDDI is built into the Microsoft .NET platform

UDDI uses World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) Internet standards such as XML, HTTP, and DNS protocols, while it uses WSDL to describe interfaces to web services. Additionally, cross platform programming features are addressed by adopting SOAP, known as XML Protocol messaging specifications found at the W3C Web site.

Any industry or businesses of all sizes can benefit from UDDI. Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Nor was there a method of how to integrate into each other's systems and processes.

Problems the UDDI specification can help to solve:

- Making it possible to discover the right business from the millions currently online
- Defining how to enable commerce once the preferred business is discovered
- Reaching new customers and increasing access to current customers
- Expanding offerings and extending market reach
- Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
- Describing services and business processes programmatically in a single, open, and secure environment

If the industry published an UDDI standard for flight rate checking and reservation, airlines could register their services into an UDDI directory. Travel agencies could then search the UDDI directory to find the airline's reservation interface. When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface.

UDDI is a cross-industry effort driven by all major platform and software providers like Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, and Sun, as well as a large community of marketplace operators, and e-business leaders[17]. Over 220 companies are members of the UDDI community.

4.3.5 Using WSDL in a UDDI Registry

The UDDI data structures[18] provide a framework for the description of basic business and service information, and architects an extensible mechanism to provide detailed service access information using any standard description language. Many such languages exist in specific industry domains and at different levels of the protocol stack. The Web Services Description Language is a general purpose XML language for describing the interface, protocol bindings and the deployment details of network services. WSDL complements the UDDI standard by providing a uniform way of describing the abstract interface and protocol bindings of arbitrary network services.

To better understand the use of WSDL in a UDDI registry you can refer for example to [8] or [9]. In these documents the implementation process is described with more details and examples.

The two UDDI data structures that are particularly relevant to the use of WSDL in the context of a UDDI registry are:

1. the tModel, also known as the service type definition
2. the businessService

tModels

A tModel provides the ability to describe compliance with a specification, a concept, or a shared design. tModels have various uses in the UDDI registry. We are interested here in the use of tModels to represent technical specifications like wire protocols, interchange formats and sequencing rules. When a particular specification is registered with the UDDI repository as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification.

The businessService

Services are represented in UDDI by the businessService data structure, and the details of how and where the service is accessed are provided by one or more nested bindingTemplate structures.

A WSDL service interface description is then published as UDDI tModel, and a WSDL service implementation description is published as a UDDI businessService.

A bindingTemplate specifies a network endpoint address (in the *accessPoint* element) and a stack of tModels describing the service.

```
<businessService>
   (…)
   <bindingTemplates>
```

[17] More information about UDDI is available at http://www.uddi.org/about.html
[18] http://www.w3schools.com/wsdl/wsdl_uddi.asp

```
      <bindingTemplate>
         (…)
         <accessPoint urlType="http"> http://www.etc.com/</accessPoint>
         <tModelnstanceDetails>
            <tModelnstanceInfo tModelKey="…">

            </tModelnstanceInfo>
            (…)
         </tModelnstanceDetails>
      </bindingTemplate>
      (…)

   </bindingTemplates>
</businessService>
```

Listing 4.3 – Sample of a BindingTemplate

WSDL service descriptions can be structured in multiple ways. However, if the reusable information is separated from the information that is specific to a given service instance, the use of WSDL and UDDI together becomes particularly simple.

4.3.6 Using BPEL4WS in a UDDI registry

WSDL describes the static interface of Web services, which includes definitions of individual operations. This may be adequate for Web services participating in stateless message exchanges. For Web services, which participate in longer conversations, it is necessary to describe the behaviour of the services in terms of dependencies, either logical or temporal, among exchanged messages. This is the focus of several efforts including BPEL4WS, now under standardization by the OASIS WSBPEL TC.

BPEL4WS abstract processes complement abstract WSDL interfaces describing behavioural aspects of Web services and providing data needed for integration with business partners. Abstract processes are used to specify the order in which business partners may invoke operations. Therefore it may be also of interest to exchange abstract processes between business partners. Software companies and standards bodies may use a UDDI registry to publish different types of services and business users may populate the registry with descriptions of services they support.

More information on WSDL[19] can be found at the W3C web site.

4.4 Service Metadata Testing

Testing a Web Service sometimes requires an overall test that also includes the service metadata information. Metadata is used to semantically describe other information. With this meaning a web service is described by a metadata schema and testing a service also involves the testing of the underlying schema.

Document [0] has covered the metadata topic with respect to the INSPIRE programme and the survey has given an exhaustive picture of some of the software solutions available on the market either free or with fee.

Different levels of requirements have been expressed by INSPIRE and concerning the availability of metadata elements related to spatial resources:

- Metadata on metadata, to know the status of metadata itself.

[19] http://www.w3.org/TR/wsdl

- Discovery metadata, composed of two levels. *Level 1* ensures the look-up for both expert and non-expert users, while *Level 2* is detailed enough for high-level discovery of spatial resources by expert users only.
- Evaluation metadata includes Level 2 metadata and a set of metadata elements needed to evaluate the fitness for use of the spatial resource.
- Use metadata

Different products can suit the need to test a metadata schema for conformance with a given standard. Some of this products have been described in Chapter 5 of [0] and one of the most interesting ones is almost surely the freeware ISO Metadata Editor[20] maintained by the Spanish National Institute for Aerospace Technology (INTA).

The main drawback of the evaluated software, pointed out in the critical review of Chapter 6, is the somewhat limited flexibility to adapt to different standards other than the ones for which the solution has been developed. In the few cases where the user can modify and maintain the configuration data required to validate a given schema, the changes always require a high skill due to the complex information underlying a metadata schema: the use must first know and then define the schema structure required by the validation process.

The survey has proved anyway that users can find different software solutions that can address the main requirements related to management (validation and interoperability) of metadata resources.

[20] This solution is available at the following link: http://www.crepad.rcanaria.es/metadata/en/index_en.htm

5. Conformance

Analysing the existing solutions available on the market, either costly or not, is too large and time expensive task to be carried out within the limited scope of this survey.

Just to give a picture of the present situation, the following list reports the wide spectrum of software solutions and market classifications available today:

- Web Services suites
  - BEA Systems, Inc. (BEA AquaLogic)
  - Cordys (Cordys Platform)
  - iWay Software (iWay Integration Solutions)
  - Kenamea, Inc. (Composite Application Suite)
  - Magic Software Enterprises, Ltd. (iBOLT Integration Suite)
  - Novell, Inc. (Novell exteNd Composer)
  - Progress Software Corporation (Actional, Sonic ESB)
  - ReadiMinds Systems & Services Pte Ltd. (ReadiMinds WebServices Applications Suite – WSS)
  - Software AG (crossvision)
  - Systinet Corporation (Systinet Product Suite)
- Web Services desktop integration
  - NetEdge Software, Inc. (Web Services Enabler)
  - RatchetSoft, LLC (Ratchet-X)
- Web Services development tools
  - Above All Software, Inc. (Above All Studio)
  - Altova, Inc. (Altova MissionKit for XML Developers)
  - Ascential Software Corporation. (Enterprise Integration Suite)
  - Attachmate Corporation (AttachmateWRQ Verastream)
  - AZORA Technologies, Inc. (AZORA SOA Studio Pro)
  - BEA Systems, Inc. (BEA WebLogic Workshop)
  - Brunswick Corporation (Redberri)
  - Compuware Corporation (OptimalJ)
  - eviware.com (soapui)
  - FusionWare Corporation (FusionWare Integration Server)
  - GT Software, Inc. (Ivory Service Architect)
  - Infologica Pty Ltd. (Infologica Web Services Framework)
  - InterSystems Corporation (Ensemble)
  - IONA Technologies (Artix)
  - Orinda Software Ltd. (OrindaBuild)
  - Pantero Corporation (Shared Data Services Suite)
  - Rogue Wave Software, Inc. (Lightweight Enterprise Integration Framework – LEIF)
  - Seagull Software Systems, Inc. (LegaSuite)
  - SeeBeyond, Technology Corporation (The SeeBeyond Integrated Composite Application Network – ICAN – Suite – Acquired by Sun Microsystems)

- StrikeIron, Inc. (StrikeIron)
- TIBCO Software, Inc. (TIBCO BusinessWorks)
- WebCollage, Inc. (WebCollage Syndicator)
- webMethods Inc. (webMethods Glue)

➢ Web services repository
- GridScope, Inc. (GridScope Repository)
- Infravio, Inc. (X-Registry)

➢ **Web services testing**
- **Borland Software Corporation (SilkPerformer)**
- **iTKO Corporation (LISA)**
- **Mindreef, Inc. (SOAPscope)**
- **Parasoft Corporation (SOAtest)**
- **SOASTA, Inc. (SOASTA Concerto)**

➢ Web Services management
- AmberPoint, Inc. (AmberPoint management solutions)
- Blue Titan Software, Inc. (Network Director and Network Director RM)
- Computer Associates International, Inc. (Unicenter Web Services Distributed Management – WSDM)
- Confluent Software, Inc. (Confluent Web Services Management Platform)
- GridScope, Inc. (GridScope Console)
- Infravio, Inc. (AppManager for Web Services)
- itellix Software Solutions (Wisiba)
- Managed Methods, Inc. (JaxView and JaxView Enterprise)
- Progress Software Corporation (Actional SOAPstation)
- SOA Software, Inc. (Service Manager and Registry)
- webMethods Inc. (webMethods Fabric)
- WestGlobal (mScape)

➢ Web Services monitoring tools
- Computer Associates International, Inc. (Unicenter Web Services Distributed Management – WSDM)
- Compuware Corporation (Vantage)
- GridScope, Inc. (GridScope Console)
- Managed Methods, Inc. (JaxView and JaxView Enterprise)
- Mercury Interactive Corporation (SiteScope)
- Mindreef, Inc. (SOAPscope)
- TIBCO Software, Inc. (TIBCO BusinessWorks)

➢ Web Services networks
- Accordare, Inc. (Reflector)

➢ Web Services orchestration
- ActiveBPEL, LLC. (ActiveBPEL)
- Active Endpoints, Inc. (ActiveWebflow)
- Corticon Technologies, Inc. (Corticon Decision Management Platform)
- Dralasoft, Inc. (Dralasoft Workflow)

- IBM Corporation (WebSphere Business Integration Modeler)
- Intalio (Intalio|BPMS)
- Metastorm, Inc. (Metastorm BPM Suite)
- OpenStorm Software, Inc. (Service Orchestrator)
- Oracle Corporation (Oracle BPEL Process Manager – formerly the Collaxa BPEL Server)
- Parasoft Corporation (BPEL Maestro)
- PNMsoft Software (Sequence BPMN/BPEL BPM Suite)
- Progress Software Corporation (Sonic Orchestration Server)
- ReadiMinds Systems & Services Pte Ltd. (ReadiMinds WebServices Applications Suite – WSS)
- SeeBeyond, Technology Corporation (eInsightBusiness Process Manager – acquired by Sun Microsystems)
- Skelta Software (Skelta Workflow.NET)
- Software AG (crossvision Service Orchestrator)
- WebV2, Inc. (WebV2 ProcessCoupler)

➢ Web Services security

- DataPower Technology, Inc. (XML Security Gateway)
- Forum Systems, Inc. (XML-Sentient)
- Layer 7 Technologies, Inc. (SecureSpan)
- Netegrity, Inc. (TransactionMinder)
- Ping Identity Corporation (PingTrust)
- Reactivity (Reactivity XML Gateway)
- RSA Security, Inc. (RSA BSAFE Secure-WS)
- SOA Software, Inc. (XML VPN)
- Teros, Inc. (Teros Web Services Security Gateway)
- TrustedWebServices.org (free collection of services and source code based on Safelayer's TrustedX WS technology)
- Vordel Limited (VordelSecure and VordelDirector)

➢ Web Services/Service-Oriented Architecture providers[21]

- Bluedog: it's a SOA and Web Services solution provider, serving government and commercial clients in the U.S. and the E.U. with a rules-engine approach.
- Prima Solutions: it has developed an insurance-specific, standards-based software foundation to design, develop, deploy, manage, monitor and maintain service-oriented insurance applications. It is built around an insurance reference model and a service repository.
- StrikeIron, Inc. (StrikeIron Web Services Business Network)
- XwebServices.com: Web Services offered include ecommerce, real estate, content management, lead management, and tools such as email validation. They also offer SOA Services such as SOA Consulting (Finance, HIPAA, and Ecommerce), SOA and Web Services Training, and SOA and Web Services Hosting.

A special mention must be made to the WS-I Test Tools Working Group[22].

[21] This is a new category and is still incomplete.
[22] http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools

This group is responsible for developing the supporting documentation and processes for WS-I Test Tool development, and the Test Materials used to test Web service implementations for conformance with a WS-I profile. Since the profiles will vary in content, the testing tools and supporting materials should be designed so that they can be easily enhanced to support new profiles while still supporting the existing profiles.

The Test Tools Architecture consists of a message monitor and an analyzer. The monitor is used to log the messages that were sent to and from a Web service. The analyzer is used to validate that the Web service interactions contained in the message log conform to a WS-I profile. The analyzer is responsible for validating all of its input artefacts. This includes the message log file from the monitor, the WSDL-based Web service description, and UDDI entries. These tools will be described more exhaustively in a dedicated sub-paragraph.

The following paragraphs analyse different software solutions, ranging from open source, free products up to market sold products. Each paragraph outlines the product functionalities with respect to web services and the discovery, definition and usability of the same.

The following software will be reviewed:

1. Apache Jmeter (Apache Foundation)
2. Apache Axis and W.S. Invocation Framework (Apache Foundation)
3. Visual Web Service Client (DigitForge)
4. Lisa WS-TestingTM (iTKO)
5. Mercury Service Test (Mercury)
6. SOA Test 4.5 (Parasoft)
7. WS-I Testing Tools (WS-I)
8. Mindreef Solutions (Mindreef)
9. Optimyz's WebServiceTester 3.0

Please Note: all the figures displayed in the next paragraphs have been taken from the products' official documentation available at the web sites specified in the footnotes.

5.1 Apache Jmeter

Apache Jmeter[23] is a desktop application implemented in Java and designed to load test functional behaviour and measure performance.

Apache Jmeter may be used to test performance both on static and dynamic resources like files, Servlets, Perl scripts, Java Objects, Data Bases and Queries or FTP Servers (and more). It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. It can also be used to make a graphical analysis of performance or to test the server/script/object behaviour under heavy concurrent load.

Apache Jmeter features include the following:

- Can load and performance test HTTP and FTP servers as well as arbitrary database queries (via JDBC)
- Complete portability and 100% Java purity.
- Full Swing and lightweight component support (precompiled JAR uses packages javax.swing.*).
- Full multithreading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
- Careful GUI design allows faster operation and more precise timings.
- Caching and offline analysis/replaying of test results.
- Highly Extensible, in detail:
    - Pluggable Samplers for unlimited testing capabilities.
    - Several load statistics with pluggable timers
    - Data analysis and visualization plugins for extendibility as well as personalization.
    - Functions to provide dynamic input to a test or provide data manipulation.
    - Scriptable Samplers (BeanShell is fully supported; and there is a sampler which supports BSF-compatible languages)

In order to use Jmeter, the user must first define a Test Plan, selecting the type of test to carry out. Different test plan models are available to users:

- Web Test Plan
- Advanced Web Test Plan
- JDBC
- FTP
- JMS Point-to-Point
- JMS Topic
- LDAP
- LDAP Extended
- Web Services (SOAP)

The plan that can better interest this survey is the Web Services Test Plan. The current implementation of the Web services sampler uses Apache SOAP driver, which requires activation.jar and mail.jar from SUN. Due to license restrictions, Jmeter does not include the jar files in the binary distribution.

[23] http://jakarta.apache.org/jmeter/index.html

5.2 Apache Axis (Apache eXtensible Interaction System)

Apache Axis[24] is an implementation of the SOAP submission to W3C. It supports the WSDL standard (with tools WSDL2Java and Java2WSDL) and uses the HTTP and JMS transport protocol.

In summary the main Axis features are:

- full support of SOAP 1.1 (and partial support for SOAP 1.2 for the present);
- support UDDI4J from IBM (but not UDDI);
- support EJB
- uses JAX-RPC Call class instead of WSDL;
- instant web service deployment (JWS) or custom deployment through a descriptor (WSDD)
- use a standalone HTTP server;
- support WSDL by means of WSDL2Java Java2WSDL and generates WSDL for deployed services;

Interoperability

The WSDL Tools subsystem contains WSDL2Java and Java2WSDL, but the Axis runtime does not.

The WSDL2Java tool takes a description of a web service written in WSDL and emits Java artefacts used to access the web service. There are three layers inside the tool:

- framework: SymbolTable, Emitter, WriterFactory
- WSDL2Java plugin to the framework: WSDL2Java (the main), JavaWriterFactory, and all the WSDL-relative writers: JavaPortTypeWriter, JavaBindingWriter, etc.
- The actual WSDL2Java emitters, one for each file generated: JavaInterfaceWriter, JavaStubWriter, etc.

Please note: for Axis 1.2, the Apache Software Foundation is focusing on their document/literal support to better address the WS-I Basic Profile 1.0 and JAX-RPC 1.1 specifications.

[24] http://ws.apache.org/axis/

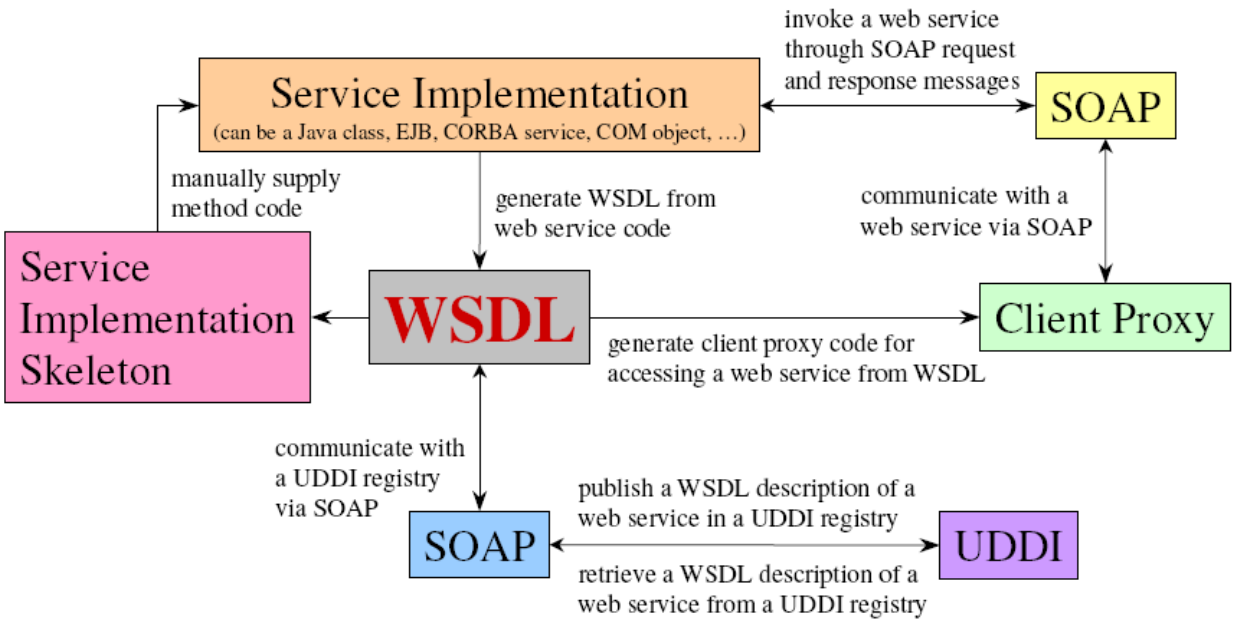

Figure 5.1 – General Web Service Toolkit Schema

The Apache Axis provider allows WSIF to invoke SOAP services via the use of Apache Axis. A detail of WSIF is given in the following of this paragraph.

Web Services Invocation Framework

The Web Services Invocation Framework (WSIF) supports a simple Java API for invoking Web services, no matter how or where the services are provided. The framework allows maximum flexibility for the invocation of any WSDL-described service.

In the WSDL specification, Web service binding descriptions are extensions to the specification. So the SOAP binding, for example, is one way to expose the abstract functionality (and there could be others). Since WSIF mirrors WSDL very closely, it also views SOAP as just one of several ways you might wish to expose your software's functionality. WSDL thus becomes a normalized description of software and WSIF is the natural client programming model.

WSIF allows stubless or completely dynamic invocation of a Web service, based upon examination of the meta-data about the service at runtime. It also allows updated implementations of a binding to be plugged into WSIF at runtime, and it allows the calling service to defer choosing a binding until runtime.

WSIF (IBM has donated WSIF to Apache Software Foundation) enables developers to interact with abstract representations of Web services through their WSDL descriptions instead of working directly with the Simple Object Access Protocol APIs, which is the usual programming model. With WSIF, developers can work with the same programming model regardless of how the Web service is implemented and accessed.

In WSDL a binding defines how to map between the abstract PortType and a real service format and protocol. For example, the SOAP binding defines the encoding style, the SOAPAction header, the namespace of the body (the targetURI), and so forth.

WSDL allows there to be multiple implementations for a Web Service, and multiple Ports that share the same PortType. In other words, WSDL allows the same interface to have bindings to, for example, SOAP and IIOP.

WSIF provides an API to allow the same client code to access any available binding. As the client code can then be written to the PortType it can be a deployment or configuration setting (or a code choice) which port and binding it uses.

WSIF let you use WSDL as a normalized description of disparate software, and allows you to access this software in a manner that is independent of protocol or location. So whether it is SOAP, an EJB, JMS (or potentially .NET and other software frameworks), APIs are centred on the WSDL used to access the functionality. This lets developers write code that adapts to changes easily. The separation of the API from the actual protocol also means flexibility: it is possible to switch protocols, location, etc. without having to even recompile the client code. So if I externally available SOAP service becomes available as an EJB, you can switch to using RMI/IIOP by just changing the service description (the WSDL), without having to make any modification in applications that use the service.

The web pages[25] dedicated to this product exposes exhaustive samples to better explain implementation within the WSIF framework.

[25] http://ws.apache.org/wsif/wsif_samples/index.html

5.3 DigitForge Visual Web Service

Visual Web Service Client[26] is an easy-to-use, interactive solution to testing and manipulating .Net and Java Web Services. VWS is one-stop shop for managing multiple web services and getting all of the information that you require from a Web Service including data structures, descriptions, SOAP request & response data, and formatted results.

According to the software manufacturer, the Visual Web Service (VWS) client is a must-have for the usual set of development tools! This utility allows you to test and execute web service methods without writing a line of code. All you need to supply is your web service location and you are up-and-running!

Refer to the DigitForge main site[27] for more information.

[26] http://www.digitforge.com/Products/VWS/Default.aspx
[27] http://www.digitforge.com

5.4 iTKO's Lisa WS-Testing™

LISA WS-Testing™[28] is a Web Services test authoring and execution tool, pulled from the full LISA SOA Testing solution, which both developers and QA/Business teams can use. This tool supports all of the current protocols and unit/functional/regression tests you need to build and launch thorough test workflows against your WSDL and SOAP objects.

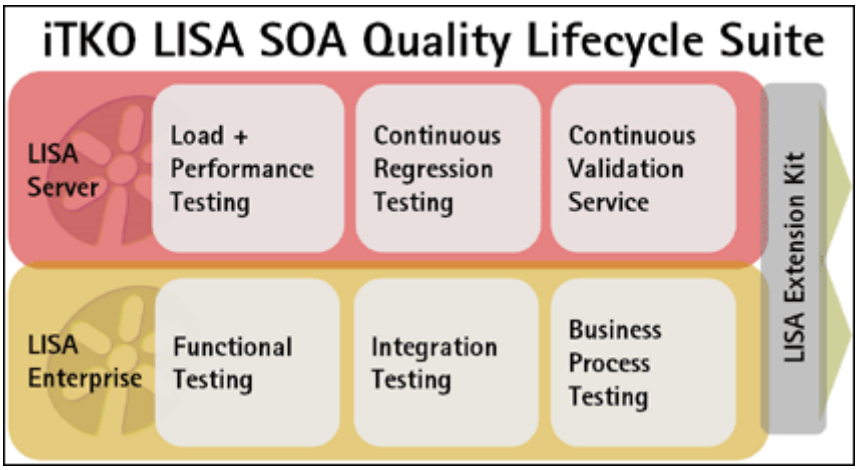

Figure 5.2 – Lisa SOA Suite

The Goal of Service Enablement

For most companies, the first strides toward a SOA strategy often involve enabling some web services on top of a few new or existing technologies. IT analysts predict that the majority of enterprise applications will be deployed in part via web services by the year 2007.

Web services allow companies to expose business logic and legacy systems as "services" that can be leveraged by multiple applications or interfaces. The ability to flexibly construct applications by connecting SOAP components means you no longer need to go through a "big bang" implementation. Multiple web sites or applications can pick and choose which services to connect to build their workflow.

The Challenge

While web services offer extensive flexibility and cost advantages, this ease of integration can come with a price. With each new web service connection you add to the mix, you create another point of failure in a business workflow. Many of the web services you need to use may be developed by other groups, or even other companies, so all of the moving parts may not be under your control.

Further, web services are used to build dynamic applications that do not come with a set user interface. It is possible to build one or more web interfaces that talk to a web service, but to directly validate that SOAP object under development teams must typically code a "test client" and maintain fragile testing scripts.

There are many tools that cover one point of failure in your web services applications – for example, unit testing of calls, examining your SOAP code, or measuring response times for requests. Most tools require some heavy development, and even have their own custom scripting language. When you try to

[28] http://www.itko.com/site/ws-testing/index.jsp

cobble these tools together into a solution, you may find yourself building and supporting your very own testing framework.

When it comes to creating a repeatable, thorough testing strategy, you need a solution that won't fail – until you need it to. LISA offers true no-code test automation to Web Services development and QA teams.

Features

LISA WS-Testing™ supports deep testing of WSDL libraries, and the SOAP objects that they generate. LISA allows you to test these "headless" components as if accessing them from multiple web interfaces or through other web services. And further, LISA lets you directly map to your services without ever developing a test client or maintaining scripts.

- No-code SOAP/XML recording/testing and WSDL exploration and test maintenance. All you need to know is the URL to capture and invoke any type of test against a web service!
- Use the same tests throughout design, development, deployment and performance/service maintenance processes of Java or .NET-based web services.
- Multiple roles. LISA is no-code automated testing, meaning developers no longer have to script tests, and non-programming team members in QA and business requirements teams can also get involved in functional testing.
- Multiple systems. One LISA test case can follow a complex workflow and validate multiple web sessions, web services, and servers. LISA supports active sessions, WS-Security protocols, authentication and magic strings, so it lets you test systems just as your end users will.

How LISA WS-Testing™ Works

iTKO's LISA WS-Testing supports both the creation and staging of Web Services tests. Developers and QA teams use a point-and-click interface to record current states and make logical "assertions" against WSDL and SOAP objects, and LISA automates many of the compliance checks and staging documents and delivers the results in an easy-to-use format. All of these steps occur in near real-time, as you are never "writing" test code to directly interact with Web Services with LISA.

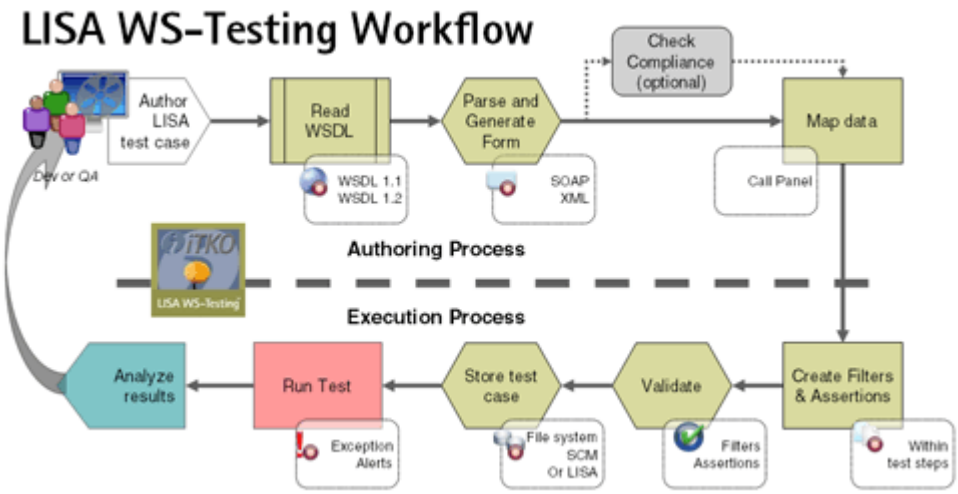

Figure 5.3 – Lisa WS-Testing Workflow

Benefits

No-code testing means less time coding tests and test clients, and more time testing. Developers and non-developers can rapidly learn and use LISA. “Live Interaction” lets you author and adjust the test as you observe live behaviour from the web service, and you can continue to execute your tests without recompiling test code.

Use LISA’s easy point-and-click testing interface, then launch LISA tests from a command line.

LISA runs on any client and supports Java and .NET and any other SOAP-compliant web services.

Since LISA WS-Testing test cases and test runs are saved as XML files, you can incorporate them easily into your process as attachments to groupware, SCM (Source Code Management) issue tracking and requirements management tools.

5.5 Mercury products

This product has been used for years by developers' team to put under test the software solutions before deploying the final release.

The Mercury products available at present on the software market can address different testing scenario and they are up to date with the latest software standards and technologies.

The Mercury mail site states that "*Mercury products and services are aligned around four optimization centres as well as two lifecycle solutions for managing application change and performance. Mercury also provides an integrated lifecycle solution for SOA governance, quality, and management*".

The present study has focused on the web-side testing capabilities of the Mercury products. In detail, the offer for the market comprises the ***Service Test*** module that enables the testing of web services implemented in a SOA framework.

With Mercury Service Test[29] it is possible to:

- Reduce QA cycle times by automating functional regression and performance testing of services that lack a GUI.
- Simulate J2EE, AXIS, and .NET client environments to test services interoperability.
- Simulate production environments through server stub simulation and asynchronous testing.
- Support the testing of multiple SOA interfaces.
- Create and track server calls to ensure successful asynchronous performance testing.
- Deliver service emulation capabilities that allow testers to get started early, before services are actually built.

Service-oriented architecture (SOA) drives business results. But SOA adds IT complexity and can cause business disruptions if not implemented correctly. Forward-thinking IT leaders are adopting a new approach to SOA and Web-services testing based on Mercury Service Test™. Mercury Service Test enables IT teams to conduct both functional and performance tests for services. Built with Mercury's industry-leading Mercury LoadRunner® technology, it can greatly reduce testing time and help ensure that services will meet the functional and performance requirements of the business before being deployed into production.

5.5.1 Mercury QuickTest Professional

Mercury QuickTest Professional™ provides the industry's best solution for functional test and regression test automation – addressing every major software application and environment. This next-generation automated testing solution deploys the concept of Keyword-driven testing to radically simplify test creation and maintenance. Unique to QuickTest Professional's Keyword-driven approach, test automation experts have full access to the underlying test and object properties, via an integrated scripting and debugging environment that is round-trip synchronized with the Keyword View.

QuickTest Professional satisfies the needs of both technical and non-technical users. It enables you to deploy higher-quality applications faster, cheaper, and with less risk. It works hand-in-hand with Mercury Business Process Testing™ to bring non-technical subject matter experts into the quality process in a meaningful way. Plus, it empowers the entire testing team to create sophisticated test suites with minimal training.

[29] http://www.mercury.com/us/products/quality-center/functional-testing/service-test/

The deployment of Mercury QuickTest Professional is optimized through the use of Mercury Best Practices. Mercury best practices cover all aspects of deployment, including product installation and operation, organizational design, process implementation, continual process improvement and measurement of return on investment (ROI). Throughout your implementation Mercury applies these best practices to your specific situation, creating world-class procedures for you that drive long-term success.

With QuickTest Professional, QA teams can achieve a number of advantages:

- Empower the entire team to create sophisticated test suites with minimal training.
- Ensure correct functionality across all environments, data sets, and business processes.
- Fully document and replicate defects for developers, enabling them to fix defects faster and meet production deadlines.
- Easily regression-test ever-changing applications and environments.
- Collaborate as a tester workgroup, by sharing automated testing assets, functions, and object repositories.
- Become a key player in enabling the organization to deliver quality products and services, and improve revenues and profitability.

The new features of QuickTest Professional

- Object repository manager: enables collaboration within tester workgroups by keeping application object data in sync. Also provides ability to merge, import/export to XML files, and add objects from application screens or meta-data.
- Robust function libraries: enables sharing of function libraries within tester workgroups.
- Intelligent pre-run validation: runs pre execution resources check automatically, notifying users of missing files or resources.
- Enhanced keyword view: drag-and-drop test steps within the *Keyword View*'s natural language environment.
- Open XML report format for test results: stores test results in an open XML format, enabling you to easily customize the reports according to your own requirements, and to integrate the test result information with other applications. Test results can now be exported to HTML.
- New IDE environment: offers a highly customizable test development environment.
- New debugger: enables testers to pin-point test errors when building and maintaining test cases.
- Keyword management: manage keywords, including turning on/off specific methods from the Keyword View.
- New application area management: leverages Mercury Business Process Testing so application area definitions can now include multiple object repositories.
- Multiple document interface for function libraries and object repositories: Allows users to concurrently open and edit multiple function libraries and Object Repositories within the QuickTest Professional interface.
- Unicode support: lets users test global, multi-language deployments of their enterprise applications.
- New environment support: supports Web services, .NET 2.0, Firefox 1.5, Netscape 8, Macromedia Flex 2, Win XP 64 bit, Internet Explorer 7, and the latest ERP/CRM applications.

5.6 Parasoft SOAtest 4.5

Parasoft SOAtest[30] is the most comprehensive tool for testing Web services. SOAtest allows users to verify all aspects of a Web service, from WSDL validation, to unit and functional testing of the client and server, to performance testing. SOAtest addresses key Web service issues such as interoperability, security, change management, and scalability. Because of its flexible nature, SOAtest is an ideal choice for development engineers and QA professionals alike, since its unit tests can be leveraged into scenario-based tests, as well as load tests, without any additional scripting or re-inventing of the wheel. By utilizing SOAtest throughout the software development lifecycle, users can prevent errors, improve quality and reliability, and accelerate the time to market for their Web service initiatives.

This product is available on both Windows and UNIX platforms, with the following comprehensive set of functionalities.

(New) Functional Testing Features

- SOA Development Governance
  - Governance of WSDLs, schemas, and SOAP messages
  - Enforcement of Standards Compliance – Schema Validity, Semantic Validity, WS-I Interoperability, Adopted WS-* Standards
  - Enforcement of Best Practices – Security, Maintainability & Reusability, Custom organization best practices
- BPEL testing
  - Automated test creation
  - BPEL 1.1 conformance verification, schema validation
  - WSDL schema validation and compliance to standards
  - Testing of BPEL Processes
  - Testing of BPEL Partners
- Test Management Solution
  - Mercury Test Director and Rational TestManager Integration
  - Enhanced Command Line Interface
  - High level and detail level HTML reports

(New) Load Testing Features

- QoS – Quality of Service estimation support
  - Apply QoS metrics to a Load Test report to analyze Load Test results and verify success/failure
  - Setup QoS metrics in Scenario configuration. Use either pre-set or create custom metrics
  - View results and manage QoS metrics in the new “QoS Report” view of the Load Test report
- Load Test report enhancements
  - Hovering over a hit with the mouse displays detailed hit information
  - Successful/failing hits are colour-coordinated green/red. Progress and report graphs use more intuitive colour maps
  - Composite reports view allows configuration of the X-axis parameter
  - New error or ‘hit’ browser for the Table part of the Detailed Report view allows navigation through large data sets (more than 300 rows)

[30] http://www.parasoft.com/jsp/products/home.jsp?product=SOAP

- Load Test Scenario definition enhancements
    - o – New Scenario graph point editor and ability to display and select individual Profile and Machine graphs to alleviate manipulation of ‘dense’ graph arrangements
    - o – Ability to globally scale Profile and Machine graphs or only those that have been selected
    - o – Ability to disable/enable Profiles in the Load Test configuration

By using Parasoft SOAtest throughout the software development lifecycle, it is possible to prevent errors early in development, improve quality and reliability, and accelerate time to market. The sooner you detect a problem, the easier it is to fix it. SOAtest provides a wide variety of tools and testing techniques that immediately and thoroughly exercise Web services and check their reliability.

5.6.1 SOAtest detailed features

SOAtest provides improved productivity and labour savings thanks to the automated generation of test cases. With this feature SOAtest saves significant time and resources as opposed to manual processes required to continuously re-create the same test cases at different points of the development process. Developers create tests to ensure the functionality of their Web services; however these tests are not leveraged in the downstream process.

SOAtest also allows test cases to be re-used, combined, and extended across teams, providing seamless transfer of knowledge and test case data within heterogeneous testing platforms. Functional and unit test cases created by development and QA can be used by the performance team to generate scenarios for use in load and performance testing, thereby eliminating the need for extensive script writing and script maintenance. SOAtest saves time and improves accuracy of test creation/execution.

By re-creating test cases on different platforms there is a significant risk of extending the time it takes to verify functionality. For example, testers that discover errors in one format may go back to the developers who then try to reproduce these errors in their own format. This rework loop is time consuming and inefficient. SOAtest is a uniform solution for Development, QA and Performance teams.

Since SOAtest can automatically generate a suite of test cases (using WSDL over HTTP), it saves the time and the efforts required to develop in-house tools for the same tasks.

Tools, such as client emulators, are developed in-house in order to facilitate functional testing; others, like load generation tools, are purchased and they require proprietary scripts to be written. The creation/usage of such tools and scripts add overhead to projects and make almost impossible to keep up with the evolution of industry standards and Web service complexity.

SOAtest has also the ability to leverage tests from unit testing through load testing thus allowing more test cases to be generated and ensuring greater coverage and quality of the service.

SOAtest ensures that all facets of Web Services, including interoperability, functionality, security, reliability, and availability, are extensively tested. SOAtest improves the breadth of Web services testing and also increases the amount of depth that is covered as well.

SOAtest provides a complete Policy Enforcement solution, enforcing policies with executable rules that can be applied to WSDLs, Schemas, SOAP messages and any other XML artefact or SOA meta-data component.

For example, SOAtest verifies schema and semantic validity for W3C and OASIS standards compliance, validates Basic Profile 1.1 for WS-I Interoperability compliance, and implements rules to enforce various other endorsed WS* Standards. In addition, SOAtest enforces compliance to best practices such as customized company guidelines, security, and maintainability and reusability.

New rules that could be required to enforce custom corporate policies and standards can be developed with SOAtest's RuleWizard utility, providing the means to bring all web services development policies and practices under policy enforcement.

WSDL validation can be considered as the first step in testing Web Services. Although WSDLs are generally created automatically by various tools, it doesn't necessarily mean that they are correct or compliant to new specifications. At the minimum, WSDLs should be checked for conformance to the WSDL schema via XML validation. Additional checks include interoperability conformance checks against WS-I's Basic Profile 1.0 and a regression test on the WSDL.

SOAtest automatically generates tests to validate the WSDL definitions, including schema verification. Additionally, a WS-I test will be created to ensure the interoperability of the business contract. Lastly, SOAtest can maintain a regression of the WSDL in memory that will be compared to the latest version.

An important feature of this product, almost unique for the category of this software, is the ability to apply test cases to solutions that expose the emerging BPEL technology. This feature is therefore very important in the light of interoperability of business services.

A non-trivial BPEL process can use multiple business partner links and reference multiple WSDL files that describe external resources. The number of BPEL process components that need to be tested can grow quickly as the complexity of the project increases. This makes manual testing of such processes extremely time consuming and inefficient. The fact that BPEL and Web Services technologies are new, complex, and evolving makes the task of testing BPEL processes even harder.

SOAtest allow users to automatically create taste cases from vendor-specific BPEL deployment artefacts and arranging these test cases into suites that reflect different aspects of testing of a BPEL process. Tests generated fall into four categories:

- BPEL semantic tests
- WSDL tests
- BPEL process tests
- BPEL Partner tests

Regarding the security issue, more and more vital in a fast evolving scenario, SOAtest includes security support for testing web services with security layers. At the transport level, SOAtest supports SSL (both server and client authentication), basic authentication, and cookies for session management. At the wire level, SOAtest allows customizable and highly configurable SOAP Headers for all the possible WSSecurity mechanisms employed. SOAtest has GUI forms for configuring X509, SAML, and Username security tokens; it also allows SOAP encryption and signing according to XML Encryption and XML Digital Signature specs.

SOAtest is also the only Web services solution that automatically creates security penetration tests that dynamically exercises and scans the Web service for security vulnerabilities. By testing the Web service with penetration attacks and analyzing the responses, security vulnerabilities can be discovered and fixed earlier in the software development cycle. SOAtest supports the following penetration tests: Parameter fuzzing, SQL injections, Xpath injections, XML bombs, external entities, malformed XML, invalid XML, username harvesting, large XML.

More features less central for this study are, for example, the product capabilities to perform unit tests, regression tests, functional test and load test.

More information can be found on the Parasoft web site[31].

[31] http://www.parasoft.com/jsp/home.jsp?itemId=0

5.7 WS-I Testing Tools

It is probably unattractive to check “by hand” every rule of the Basic Profile, so the WS-I has developed conformance testing tools. The first provided tool is a Monitor and Analyzer package.

These tools are based on configuration files which allow the user to enabled/disabled rules to tests (assertion files). It is possible to define a core assertion file to check in the context of IVOA Web Services.

The WS-I Test Tools consist of two tools: the monitor and the analyzer. The monitor provides an unobtrusive way to log Web service messages using a man-in-the-middle approach. The purpose of the analyzer is to determine if a set of Web service related artifacts conform to the requirements in the WS-I Basic Profile 1.0. There are three basic types of artifacts:

- messages: the set of messages that were logged by the monitor;
- description: the service description for the Web service (including any referenced XML schema definitions), if the location of the WSDL document is available;
- discovery: the UDDI entries for a Web service, if the UDDI entries reference a WSDL-based service description;

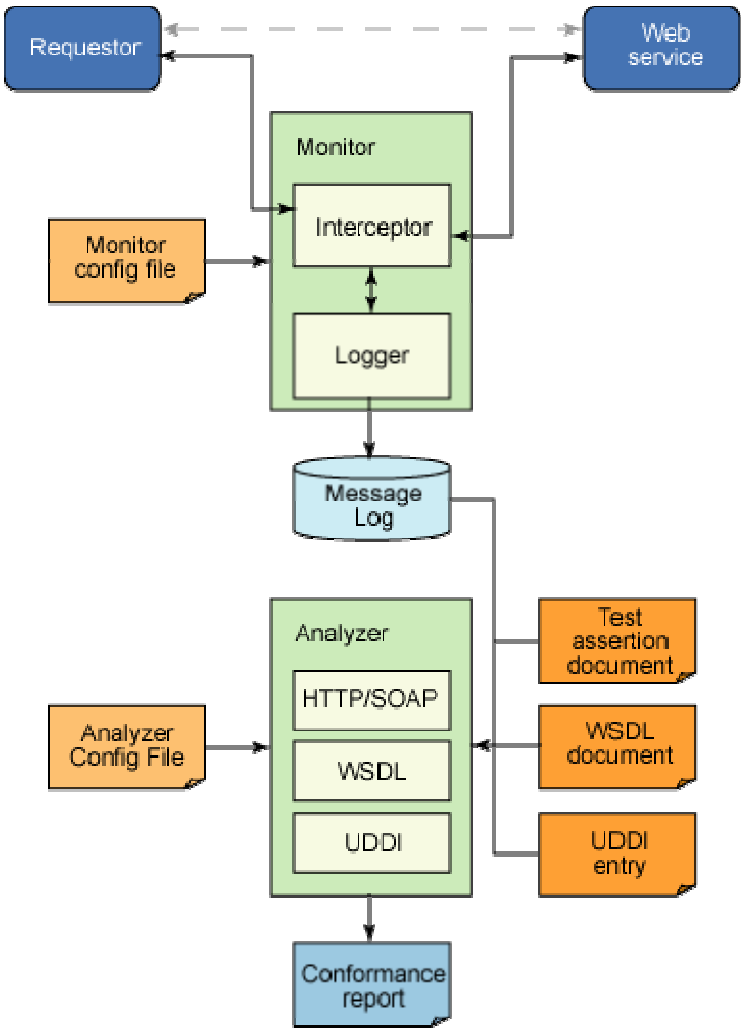

Figure 5.4 – Overview of the WS-I Test Tools architecture.

5.7.1 Monitor overview

The monitor contains two primary functions: *message interceptor* and *message logger*. The message interceptor intercepts the messages that are sent from a service requestor to a Web service and from a Web service back to the service requestor. The logger formats the intercepted messages into a standard format and then writes them out to a message log. With these two functions, a single monitor can intercept and log messages from multiple Web services. The monitor functions are controlled by a configuration file which defines the association between the ports the monitor listens on for incoming messages, and the Web service location where the monitor should forward the messages.

When using the monitor, the service requestor views it as if it was the Web service. All SOAP messages are sent to the monitor instead of the Web service. Since this is not the normal mode of operation for the requestor, there are three basic ways to do this:

- alter the requestor to point at the monitor instead of the Web service;
- move the Web service to a new location and run the monitor in its place;
- alter the Web service endpoint information that the requestor uses;

There are several system configurations that can be used to run the monitor. There are four basic system configurations which define the systems where the requestor, monitor, and Web service can run:

- the requestor, monitor, and Web service can be run on the same system;
- the monitor can be run on the same system as the requestor, and the Web service can run on a different system;
- the requestor can be on a different system than the monitor and Web service;
- the requestor, monitor, and Web service can be run on three different systems.

The monitor requires one input file. This file is an XML document that contains the configuration options that tell the monitor what it needs to monitor and where it needs to log the messages it intercepts.

5.7.2 Analyzer overview

The analyzer tool determines if the artifacts for a Web service conform to the Basic Profile by processing a set of test assertions. A test assertion is a testable expression of one or more requirements in the Basic Profile. All of the test assertions are listed in a test assertion document (see Resources), which is an XML document whose contents are segmented by artifacts type (discovery, description, and messages).

The input for the analyzer includes the location of the test assertion document and references to the Web service artifacts. The output from the analyzer is a conformance report. All of this information is specified in the analyzer configuration file.

Just like the monitor tool, the analyzer uses an XML document to define its configuration options. Listing 5.1 contains an example of an analyzer configuration file.

```
 1 <configuration name=”Sample Basic Profile Analyzer Configuration”
 2     xmlns=”http://www.ws-i.org/testing/2003/03/analyzerConfig/”>
 3   <description>
 4     This file contains a sample of the configuration file for
 5     the Basic Profile Analyzer.
 6   </description>
 7
```

```
 8   <verbose>false</verbose>
 9   <assertionResults type="all" messageEntry="true" failureMessage="true"/>
10   <reportFile replace="true" location="report.xml">
11     <addStyleSheet href="c:/wsi-test-tools/common/xsl/report.xsl" type="text/xsl"/>
12   </reportFile>
13   <testAssertionsFile>
14     c:/wsi-test-tools/common/profiles/BasicProfileTestAssertions.xml
15   </testAssertionsFile>
16   <logFile correlationType="endpoint">
17     log.xml
18   </logFile>
19   <uddiReference>
20     <uddiKey type="bindingKey">22e841c0-0ef2-11d7-a725-000629dc0a53</uddiKey>
21     <inquiryURL>http://uddi.ibm.com/ubr/inquiryapi</inquiryURL>
22   </uddiReference>
23 </configuration>
```

Listing 5.1 – Analyzer Configuration File

5.7.3 Test assertion document

Test assertions are used by the analyzer testing tool V0.96 to determine if a Web service is conformant to the Basic Profile. To view in detail the test assertions for the WS-I Basic Profile definition, please refer to [11].

Each test assertion has a unique identifier and contains all of the information that is needed to understand how the analyzer will process the assertion. In the example of Figure 5.5 the test assertion identifier is WSI2406. The purpose of the test assertion is to analyze a WSDL binding element to verify that the value of the use attribute is "literal" when it is used on the body, fault, header, and "headerfault" SOAP binding elements.

A "candidate" element is one that is to be verified for conformance. The binding of the tModel if <wsi-analyzerConfig:uddiReference> is given or the <wsi-analyzerConfig:wsdlElement> in the configuration file of the Analyzer define a candidate element for verification. Verification on an element also implies that the same verification is made for all the elements that it uses. That is, the elements it uses also become candidate elements.

Test Assertion: WSI2406

Entry Type	Test Type	Enabled	Additional Entry Types		Prerequisites	Profile Requirements	
			Message Input	WSDL Input		Target	Collateral
binding	required	true	none	none	WSI2703	R2706 R2723	R2707

Context:
For a candidate wsdl:binding element, if the use attribute is specified on the soapbind:body, soapbind:header, or soapbind:headerfault elements.

Assertion Description:
The use attribute has a value of "literal."

Failure Message:
The use attribute of a soapbind:body, soapbind:fault, soapbind:header and soapbind:headerfault does not have value of "literal."

Failure Detail Descripton:
Defective soapbind:body, soapbind:fault, soapbind:header, or soapbind:headerfault elements.

Comments:
R2707 is not the verified Req, but an assumed Req for this verification.

Figure 5.5 – Example of single Test Assertion

Verification it based on the following transitivity rules, applied recursively. For WSDL element references:

- a verification on a wsdl:port is inherited by the referenced wsdl:bindings;
- a verification on a wsdl:binding is inherited by the referenced wsdl:portTypes;
- a verification on a wsdl:portType is inherited by the referenced wsdl:operations;
- a verification on a wsdl:operation is inherited by the referenced wsdl:messages.

For UDDI references:

- a verification on a uddi:bindingTemplate is inherited by the referenced uddi:tModel;
- a verification on a uddi:tModel is inherited by the referenced wsdl:binding.

A test assertion may have one of five possible results, outlined in Figure 5.6.

passed	The test assertion was processed and it passed the stated assertion.
failed	An error was detected when processing the test assertion.
warning	The test assertion failed but the test type was "recommended."
notApplicable	Either the context conditions did not exist or a prerequisite test assertion failed
missingInput	The primary type of data was not specific as input to the analyzer or it did not exist in the specified artifact

Figure 5.6 – Example of single Test Assertion

If any of the test assertions fails, then the artifacts that were analyzed do not conform to the Basic Profile. When using the HTML view of the report, the assertion results are colour-coded in the same way that they are listed in the figure 5.6.

5.7.4 Conclusions

The WS-I tools test Web service implementations using a non-intrusive, black box approach. The tools' focus is on the interaction between a Web service and user applications. The tools can only verify the conformance of Web Service artifacts that are produced during a testing session. Some artifacts belong to the definition of the Web Service (WSDL); some others result from the observable behaviour of the Web Service at run-time. It is rather difficult to test all possible behaviours that a Web Service can exhibit, mostly because exercising these behaviours is application-dependent and requires an application-level understanding of the Web Service. For these reasons, the Testing WS-I working group has not attempted to provide certification criteria. Using certification criteria that are too general or incomplete will not guarantee interoperability for every use case.

The tools can also be used at development time, to verify that Web Service definitions are profile-conforming. The testing tools are then an indicator of conformance of a Web Service to the Basic Profile, based on the artifacts produced.

In turn, this is an indicator of interoperability with other business partners: by addressing requirements that concern the run-time interaction between a Web Service and another party, the tools provide a

powerful indicator of the ability of this Web Service to interoperate with any external party known to also comply with the Basic Profile.

The conformance testing-tools have been experimented for Tomcat/Axis and .NET in the context of the VO.

Please note

The WS-I Testing Tools Analyzer can be run directly from SOAPscope on messages and WSDL documents in the SOAPscope database. This is useful if you need to create a Profile Conformance Report from the WS-I Testing Tools Analyzer to document a compliance claim as part of your release cycle. SOAPscope greatly reduces the time needed to get up and running with the WS-I tools. SOAPscope seamlessly handles message capture, without using the WS-I Monitor. There is no need to create configuration files, or modify WSDL documents to make them correlate to captured messages.

SOAPscope automatically installs the Java implementation of the WS-I Testing Tools. Users must choose the messages or WSDL they wish to analyze, and run the WS-I Analyzer with just a few mouse clicks. After a Profile Compliance Report is generated, they can print it, save it, or view it in a new window.

5.8 Mindreef Solutions for Web Services and SOA

Mindreef[32] provides tools and solutions for automated testing and debugging of web services and SOA projects. From entire team collaboration throughout the service lifecycle, to individual developers and diagnosing XML problems, Mindreef solutions enable quality throughout the evolution of a SOA, starting with the initial Web service project.

Building, testing, and maintaining effective services and composite applications requires that entire project teams have access to service problems as they arise and can respond to those problems in different ways. The Mindreef product family is an award-winning set of diagnostic and testing tools that help analysts, architects, developers, testers, support engineers, consultants and managers understand and address services problems quickly and cost-effectively.

The product that can best suit our needs within the scope of this research is *SOAPscope* family, now at version 6.0[33]. This group of products include a server version, aimed to team scenarios, a workstation version for independent works and more tools, like *Load Check* and *Policy Rules Manager*.

With *SOAPscope Server*, multiple teams can leverage robust features while collaborating and sharing test assets to improve overall SOA quality, including Web services testing and QA, design-time support, prototyping, change-time support, run-time support, load testing and policy compliance validation.

SOAPscope Workstation is intended for professional developers and testers working independently, on small teams, or working remotely from a team, who do not need collaboration and sharing of service assets, Mindreef offers Mindreef SOAPscope WorkstationTM. SOAPscope Workstation offers capabilities similar to SOAPscope Server for creating, building, maintaining and supporting Web services and SOAs, with always-on tools for testing, diagnostics, governance, and support.

A special mention must be made for the *Load Check* product that can be used to test the performance and scalability of services after the development stage. According to the product advertising page, "*Mindreef Load Check* reduces the technical complexity and the costs associated with performance and load checking; it allows project teams – developers and QA testers – to perform comprehensive load testing on Web services early and often throughout the development lifecycle:

Developers and testers can use *Load Check* to:

- Create and run scenario tests against a particular Web service endpoint or set of endpoints.
- Create actions that represent the steps in a test scenario.
- Import scenario tests and specify the parameters of the test (i.e., test run time, user ramp rate etc.).

Load Check provides visibility – across the project and management team – to critical problem areas that require early attention before they are entrenched into to the architecture. In detail:

- It displays load tests results in a real-time chart during execution, making it easy to monitor.
- Test result reports summarizing the results available immediately after the test completes.
- Reports can also be generated in PDF format, for sharing with the entire team.

[32] http://home.mindreef.com/
[33] http://home.mindreef.com/index.php?option=com_frontpage&Itemid=1

5.9 Optimyz's WebServiceTester 3.0

WebServiceTester is an end-to-end solution for Web Services Testing and Diagnostics needs. It is a comprehensive integrated solution that offers Automatic Test Generation, Functional, Regression & Load Testing, Conformance Testing against WS-I Profiles, BPEL based Orchestration Testing, Secure Web Services Testing and Web Services Debugging & Diagnostics. It is a multi-platform, user-friendly, fault tolerant, scalable Web Services Testing Solution.

WebServiceTester eliminates the test development and maintenance efforts for Web Services projects so that the QA engineers can focus on Testing the Business Process and various test data points rather than focusing on programming, scripting and maintenance.

Unlike most SOAP message level testing tools, Optimyz's WebServiceTester allows the user to run tests for multiple operations within multiple WSDLs and multiple test data sets without manual intervention. It enables the user to run thousands of Web Services Tests and avoids the painful, manual and serial process of invoking Web Services and verifying the resulting SOAP responses one at a time.

5.9.1 Testing capabilities

WebServiceTester 3.0 has enhanced its BPEL based Web Services Orchestration testing by providing Load Testing capabilities for the Workflow of Web Services and easy to debug and diagnose the failures and bottlenecks. Load Testing the complete Business Process instead of simply loading individual endpoints or operations enables the QA engineers to perform scalability testing just like the real end yours would consume the Web Services. They will not need to guess and manually configure different level of Load for different operations but rather Load Test the entire Business Process.

WebServiceTester 3.0 has vastly improved support for testing secure web services. It includes support for testing web services that require XML signatures and encryption (generate signed/encrypted SOAP requests), NTLM authentication, SOAP Authentication and SSL/HTTPS.

WebServiceTester 3.0 adds support for WS-I Basic profile conformance, which would enable the user to check for WSDL compliance and SOAP message compliance. It has further added support for Debugging and Diagnostics capabilities to eliminate need for separate tools for just debugging.

WebServiceTester 3.0 has enhanced its Load Testing capabilities with by adding Server status monitoring, histograms to verify the frequency distribution of the SOAPmessages against the Service Level Agreements, improved fault safety and comprehensive reporting.

Please Note: the author was unable to track down the product web page, since there are different references to this product but the link to the producer site[34] seems to be wrong.

[34] http://www.optimyz.com/servicetester-features.html

6. Critical Review

The solutions analysed in the previous chapter cover most of the issues addressed to by the Inspire directive. These products are anyway more focused on the discovery and invoking (interoperability) services required by the Inspire work programme rather than download and transformation services.

Most of these products aim to a specific audience of the market, which are the developing teams and producers of software that exposes web (network) services or that it is tightly related to the web services technologies. Most of these products specifically address the development and subsequent testing stages of a solution. The main reason can be found in the fact that it's rather difficult to test all possible behaviours that a Web Service can exhibit, mostly because exercising these behaviours is application-dependent and requires an application-level understanding of the Web Service.

Even if these products lacks in some features and are somewhat limited in the application areas specifically correlated to the INSPIRE requirements, the scenario made up by web technologies is continuously evolving, thus allowing for new products and solution that can meet the INSPIRE conditions.

The feeling of the author, according to the results of his survey, is the likely fragmentation of the world market in order to fulfil different needs that rise in distinct organisation and geographical areas. For example, the American market and the European market operate with different needs and they can take diverging path in the exploitation of technologies and strategies of the web service domain. The standards present over the web already point out to this gap.

A second limit outlined by results of this survey is related to the technologies adopted by the analysed software solutions. Even if few are platform-free, some of them operate only within a Java environment while others with Microsoft .Net. This dependency can reveal as a real restriction, when not even an obstacle, to the testing capabilities for interoperability of exposed web services.

Take for example the Microsoft .Net framework, too much tied to the Microsoft Windows operating system while Java is rather ubiquitous across many disparate operating systems and environments[35]. Both Java, through its Java EE (aka J2EE) platform, and .NET through ASP.NET, compete to create web-based dynamic content and applications. This competition, even if positively contribution to the growth of web technologies, on the other hand introduces fragmentation whenever a single approach is required to assure conformance and interoperability.

A further weak point detected in the previous analysis, out of any doubt, is the security issue. The web service demand on the web is becoming more and more urgent and fast. The main effect of this fast-changing scenario is the growing complexity of web services from a business perspective. This emerging trend on one side requires an improvement in the efficiency of processes, while on the other side calls for a tight security. Security therefore is (and is going to be) a must both for service providers and consumers. And security also affects the way a conformance and interoperability test is carried out.

6.1 Regarding SOA Testing

A big problem with SOA testing is that although a test harness may help to ensure that each component behaves properly functionally, the linkage of many disparate components over different locations and environments can throw up really nasty performance/tuning issues and these can be real tough to resolve. In particular, if the design of the SOA service, for example, is at the wrong

[35] Common Language Runtime (CLR) is also a cross-platform standard. Even if the main target is still Windows, implementations do exist for other platforms, most notably the Mono Project.

granularity it will never deliver optimal performance no matter how well the components are developed.

SOA testing differs in significant ways from traditional testing; as highlighted in the following:

1. SOA is an architecture, not just web services. Therefore, testing has to validate across multiple industry and vendor protocols typical of most composite systems. The visibility required for error detection and root cause analysis must be native to each of the protocols.
2. SOA aims to facilitate business not just build applications. The focus is on business process. Therefore, testing must validate the process end-to-end. The ability to build these processes across the SOA infrastructures inclusive of multiple messaging technologies, ESBs, web services and legacy systems has created new "black boxes". So testing methodologies must be able to address this complexity and understand how to integrate multiple tools and processes or standardize.
3. End-to-end testing needs to be done incrementally and without delays to deliver speed to market. A unique problem with integration and SOA projects is that the testing of complete processes often requires access to multiple processes that are unavailable or not under your control (third party). Being able to simulate and maintain these simulation tests as part of your test repository is the key to test readiness and meeting project deadlines.
4. Automation is the key. SOA compels constant reuse and change. Governance and quality standards will require new standards of coverage and error rates. It all adds up to more testing, of similar components but in different combinations and in the hands of a greater number of consumers. Automated testing assures ease of use, consistent testing, management of test assets to match SOA assets, documentation of testing and results, and tremendous reduction of time and effort.

Therefore, testing SOA goes well beyond testing Web Services. To quote the OASIS consortium web site, "Web services can be used to implement SOA, but service orientation does not necessitate the use of Web services protocols, nor does the use of Web services protocols ensure that the overall system is SOA." In other words, SOA implementations can benefit from the use of web services, but it is not a given that one always comes with the other. You need to be able to test natively and intuitively at the business logic layers, something that a lot of purported SOA testing tools fail to do.

Like a SOA reference model, a SOA testing model is required to fix this situation.

6.2 Regarding BPEL Testing

Orchestration is a capability that will increasingly be present in the development scenario. Since orchestration enables fine-grained Web services to function as a larger system, an effective Web services testing tool needs to support Web services orchestration and workflow testing based on BPEL4WS, which allows companies to describe business processes that include multiple Web services and standardize message exchange internally and between partners.

One of the products that can directly address the intricacies presented by the Web services testing process, including standards conformance and orchestration, is WebServiceTester. For the most part, most testing tools are too one dimensional, tied to inflexible scripts, and built for client-to-server type applications, rather than the complexities of Web services frameworks.

There could be orchestration of Web services. There could be cascading of Web services. Or, one Web service may be calling another. There could be significant real-world business software dependencies between Web services, such as output of one Web service passed out into the other Web service, and so forth, without any GUI being involved.

Many tools look mainly at the GUI layer. This is sometime adequate, but needlessly complicates the testing job in a Web services environment, where links to back-end applications can change on the fly. The fundamental problem is that the web applications always keep changing. A good Web service-testing tool needs to be able to test SOAP messages, WSDL metadata, and, above all, interactions. Few tools anyway can provide a testing environment that covers all the bases involved in sophisticated Web services infrastructures.

7. Test Cases

For the required test cases the author has chosen a subset of the reviewed solutions out of the ones described, since most of them are not free or shareware and for the base knowledge required to use them. The next paragraphs will present some of the possible testing scenarios. The figures displayed in the next paragraphs have been taken either from the products' official documentation, available at the related web sites, or directly as screenshots of the applications.

To begin with, a very interesting Web site that can be used for testing is Xmethods[36]. This site lists publicly available web services and gives access to the following programmatic interfaces:

- Xmethods SOAP Interfaces
- Xmethods UDDI Private Registry
- WS-Inspection document
- DISCO document
- RSS feed

The site also exposes a list of common implementations and the related publishers and finally there are a fairly good number of tutorials to assist the development and testing of Web services.

In the main page there is a "Try It" link for each available web service. This feature let users to interact with the web service via Mindreef's *Scope-it?* online tools. The service's link opens a new pop-up window where users can invoke the service and inspect its structure with more details using *Scope-it*. Unfortunately, at the time this document was being drafted, the service was down for maintenance and the author could not have any opportunity to try it.

7.1 Apache Jmeter

Apache Jmeter requires a fully compliant JVM 1.3 or higher, even if it performs best with 1.4 or better. Since Jmeter uses only standard Java APIs there may be compatibility problems: JRE may fail to run Jmeter because of JRE implementation issues.

Finally, Java 1.3 does not include SSL (HTTPS) support therefore it will be required to download JSSE if Jmeter will be used to test a SSL-encrypted web server. Also, it does not perform as well as later Java versions. It is recommended to use the latest version of Java 1.4 or 1.5 for best results.

7.1.1 Elements of a Test Plan

The following elements make up a test plan:

- Thread group
- Controllers
- Samplers
- Listeners
- Assertions

[36] http://www.xmethods.net/ve2/index.po;jsessionid=Y9ARc7Vc-f0UHIGW_YOeygRb

Thread group elements are the beginning points of any test plan. All elements of a test plan must be under a thread group. The thread group element controls the number of threads Jmeter will use to execute your test. The controls for a thread group allow to:

- Set the number of threads
- Set the ramp-up period
- Set the number of times to execute the test

Each thread executes the test plan in its entirety and completely independently of other test threads. Multiple threads are used to simulate concurrent connections to the server application.

Jmeter has two types of Controllers: Samplers and Logical Controllers.

Samplers tell Jmeter to send requests to a server. For example, adding the HTTP Request Sampler make Jmeter send an HTTP request. A request can also be customised by adding one or more Configuration Elements to a Sampler. The following samplers are available:

- FTP Request
- HTTP Request
- JDBC Request
- Java object request
- LDAP Request
- SOAP/XML-RPC Request
- WebService (SOAP) Request

Logic Controllers let users to customize the logic that Jmeter uses to decide when to send requests. For example, it is possible to add an Interleave Logic Controller to alternate between two HTTP Request Samplers. A logic controller may have as child elements any of the following:

- Samplers (requests)
- Configuration Elements
- other Logic Controllers

Listeners provide access to the information Jmeter gathers about the test cases while Jmeter runs. The Graph Results listener plots the response times on a graph. The “View Results Tree” Listener shows details of sampler requests and responses, and can display basic HTML and XML representations of the response. Other listeners provide summary or aggregation information. Additionally, listeners can direct the data they collect to a file for later use. Every listener in Jmeter provides a field to indicate the file to store data to. Listeners can be added anywhere in the test. They will collect data only from elements at or below their level.

Assertions allow users to assert facts about responses received from the server being tested. Using an assertion, it is possible to “test” that an application is returning the expected results. For instance, you can assert that the response to a query will contain some particular text. The text specified can be a Perl-style regular expression indicating that the response is to contain the text, or that it should match the whole response.

A Pre-Processor executes some action prior to a Sampler Request being made. If a Pre-Processor is attached to a Sampler element, then it will execute just prior to that sampler element running.

Symmetric to the pre-processor, a Post-Processor executes some action after a Sampler Request has been made.

The following example displays a possible test tree with the plan elements:

- Test Plan
  - Thread Group
    - Once Only Controller
      - Login Request (an HTTP Request)
    - Load Search Page (HTTP Sampler)
    - Interleave Controller
      - Search “A” (HTTP Sampler)
      - Search “B” (HTTP Sampler)
      - HTTP default request (Configuration Element)
    - HTTP default request (Configuration Element)
    - Cookie Manager (Configuration Element)

Definition of a Test Plan to test a WebService

To test a Web service the following elements are required: Thread Group, WebService(SOAP) Request (Beta Code), and Graph Results.

The webservices sampler uses Apache SOAP driver, which requires activation.jar and mail.jar from SUN. Due to license restrictions, Jmeter does not include the jar files in the binary distribution.

The first step with every Jmeter Test Plan is to add a Thread Group element in order to set the number of users you want to simulate, how often the users should send requests, and the how many requests they should send.

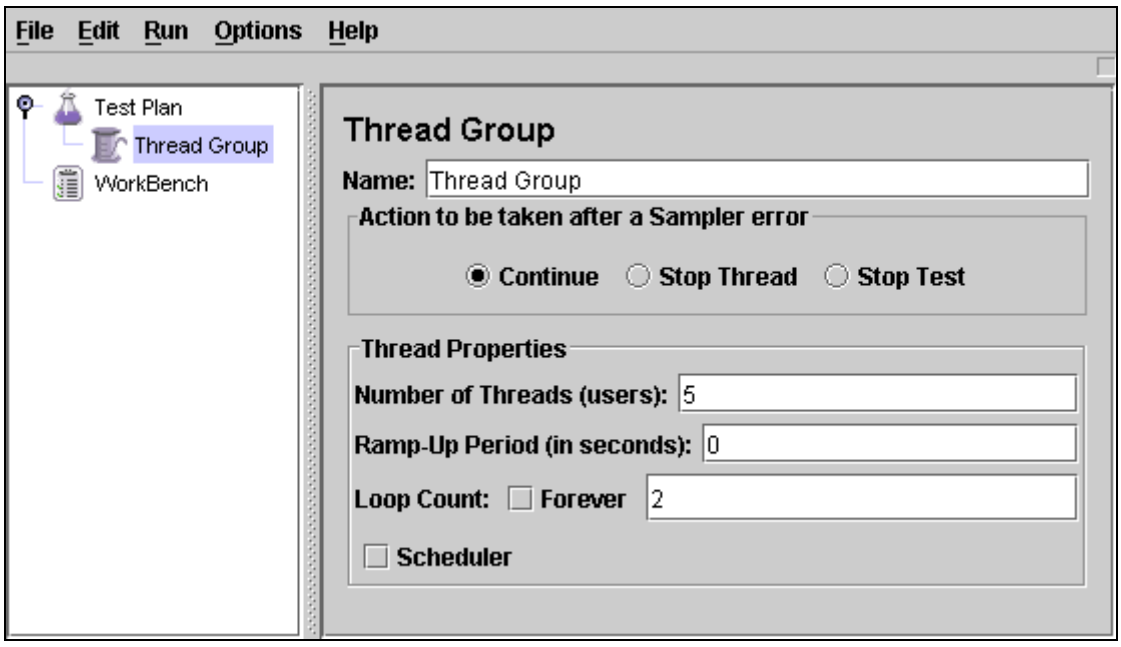

Figure 7.1 – Definition of a Thread Group

Then a sampler must be added to the test plan. In this sample, a .NET webservice will be used. Note that there is a significant difference between how .NET and Java implement webservices. This difference affects the configuration data required to make the sampler run.

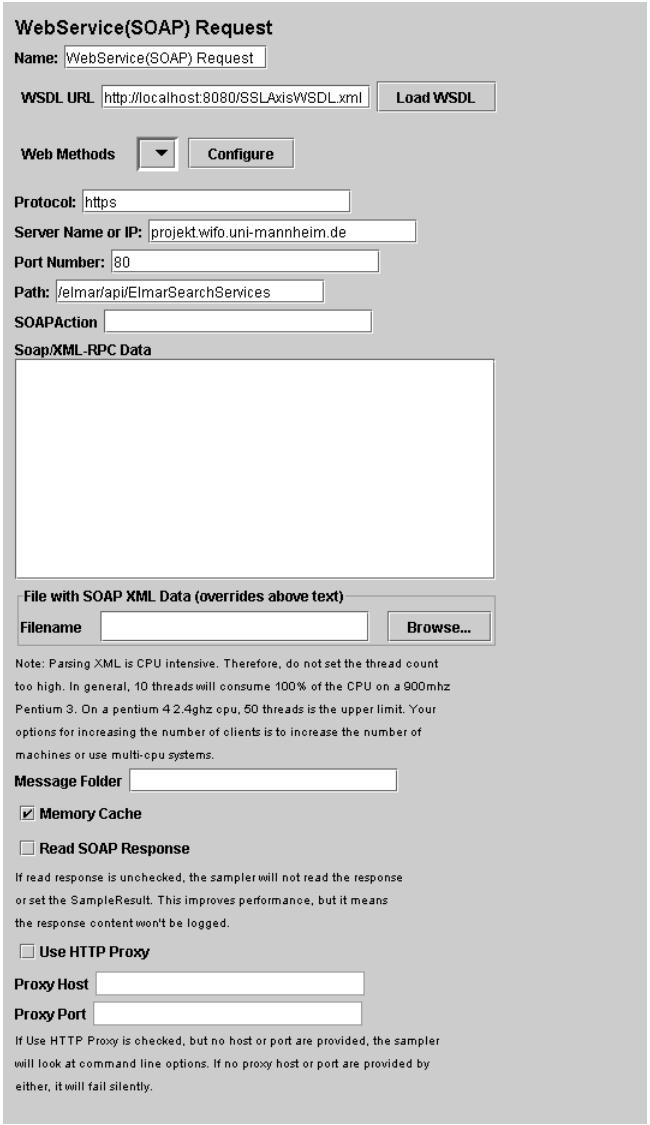

Figure 7.2 – Configuration of a Webservice Sampler

If the WSDL file is loaded correctly, the "Web Methods" drop down gets populated otherwise there is a problem getting the WSDL. The WSDL can be tested using a browser that reads XML. For example, when testing an IIS webservice the URL will look like the following address:

http://localhost/myWebService/Service.asmx?WSDL

At this point, SOAPAction, URL and SOAP Data should be blank.

Once selected the web method, this must be configured. The sampler should populate the "URL" and "SOAPAction" text fields[37]. Assuming the WSDL is valid, the correct soap action should be entered.

The last step is to paste the SOAP message in the "SOAP/XML-RPC Data" text area. It is optionally possible to save the soap message to a file and browse to the location. For convenience, there is a third option of using a message folder. The sampler will randomly select files from a given folder and use the text for the soap message.

The final element required in the Test Plan is a Listener. This element is responsible for storing all of the results of your HTTP requests in a file and presenting a visual model of the data.

[37] Currently, only .NET uses SOAPAction that will be blank for all other webservices (JWSDP, Weblogic, Axis, The Mind Electric Glue, and gSoap).

This example selects the Jakarta Users element and adds a Graph Results listener (Add → Listener → Graph Results). Next, users must specify a directory and filename of the output file, either typing it into the filename field, or selecting it with the Browse button. This last action browses to a directory where users can select the filename.

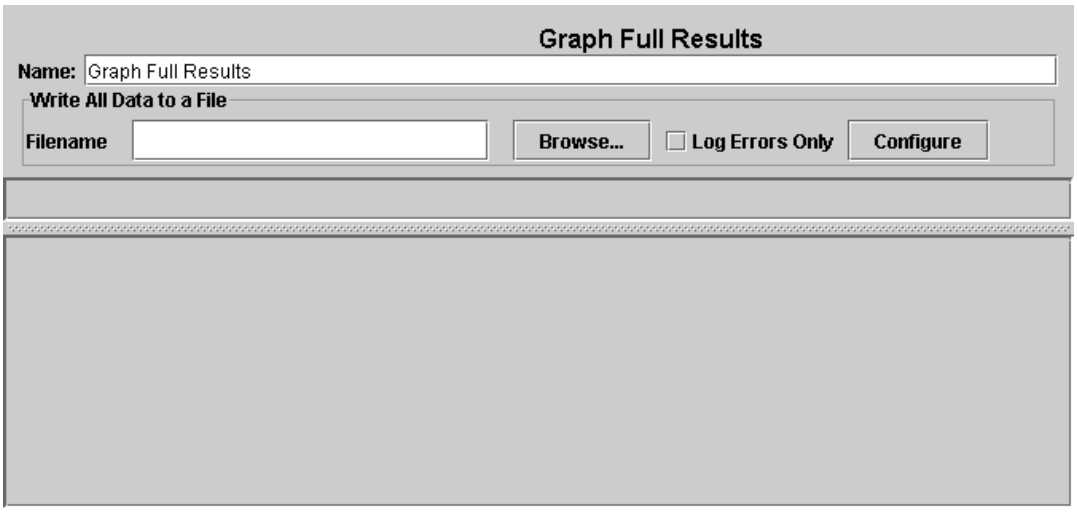

Figure 7.3 – Configuration of Output Results

To complete the Test Plan the user must save all these settings and thereafter he can execute the plan. When the execution ends, if the user has configured the listener to save the results in a file, he will be able to open that file in any visualizer. Each visualizer will display the results in its proprietary fashion. It is possible to have the same file open in more than one visualizer. Jmeter will ensure during the test run that no sample is recorded to the same file more than once.

7.2 Apache Axis

This sample is part of the WSDI distribution and demonstrates the invocation of a SOAP service with minimum functionality and using only primitive schema types. The sample invokes the *Stockquote* service, a popular SOAP service that offers one port type with a single operation. That operation takes as input a stock symbol (that has schema type string) and returns a recent stock quote for that company (the stockquote has schema type float).

This sample doesn't need the deployment of the service at all; instead it makes use of the public Stockquote service developed and hosted by Xmethods.[38]

The following is a WSDL sample file with a Web Service definition.

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions name='net.xmethods.services.stockquote.StockQuote'
  targetNamespace='http://www.themindelectric.com/wsdl/net.xmethods.services.stockquote.StockQuote/'
        xmlns:tns='http://www.themindelectric.com/wsdl/net.xmethods.services.stockquote.StockQuote/'
        xmlns:electric='http://www.themindelectric.com/'
        xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
        xmlns:xsd='http://www.w3.org/2001/XMLSchema'
        xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
        xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
        xmlns='http://schemas.xmlsoap.org/wsdl/'>
```

[38] http://www.xmethods.net/ve2/ViewListing.po?serviceid=2

```
   <message name='getQuoteResponse1'>
      <part name='Result' type='xsd:float'/>
   </message>
   <message name='getQuoteRequest1'>
      <part name='symbol' type='xsd:string'/>
   </message>

   <portType name='net.xmethods.services.stockquote.StockQuotePortType'>
      <operation name='getQuote' parameterOrder='symbol'>
         <input message='tns:getQuoteRequest1''/>
         <output message='tns:getQuoteResponse1''/>
      </operation>
   </portType>

   <binding name='net.xmethods.services.stockquote.StockQuoteBinding'
            type='tns:net.xmethods.services.stockquote.StockQuotePortType'>
      <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http''/>
      <operation name='getQuote'>
         <soap:operation soapAction='urn:xmethods-delayed-quotes#getQuote''/>
         <input>
            <soap:body use='encoded'
                       namespace='urn:xmethods-delayed-quotes'
                       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/''/>
         </input>
         <output>
            <soap:body use='encoded'
                       namespace='urn:xmethods-delayed-quotes'
                       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/''/>
         </output>
      </operation>
   </binding>

   <service name='net.xmethods.services.stockquote.StockQuoteService'>
      <documentation>net.xmethods.services.stockquote.StockQuote web service</documentation>
      <port name='net.xmethods.services.stockquote.StockQuotePort'
            binding='tns:net.xmethods.services.stockquote.StockQuoteBinding'>
         <!-soap:address location='http://66.28.98.121:9090/soap'/→
         <soap:address location='http://64.124.140.30:9090/soap''/>
      </port>
   </service>
</definitions>
```

Listing 7.1 – WSDL Sample File

The next sample outline the code required to invoke the Web service dynamically using WSIF's dynamic invocation interface (DII).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><html><head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Author" content="Nirmal Mukhi">
<meta http-equiv="Content-Style-Type" content="text/css">
<title>Web Services Invocation Framework: Samples</title>
<link rel="stylesheet" href="wsif.css" type="text/css"></head>

<body alink="#0000ff" bgcolor="#ffffff" leftmargin="2" topmargin="2" marginwidth="2" marginheight="2">

<h2>
Web Services Invocation Framework:<br>
Invoking the SimpleSOAP Sample using WSIF's dynamic invocation interface</h2>
<p>After you have <a href="../../../../doc/samples.html">set up the CLASSPATH in your environment</a>,
to invoke this sample using WSIF's DII, run the DynamicInvoker class. Specify as command line arguments
the location of the WSDL file for the stockquote sample followed by the operation you wish to invoke
and the symbol for the company whose stockquote you are interested in. For example, <br><tt>java
clients.DynamicInvoker samples/simplesoap/StockquoteSOAP.wsdl getQuote IBM</tt></p>
<p>To see details of how the WSIF API is used to make invocations dynamically, take a look at the code
for the <a href="../../../clients/DynamicInvoker.java">DynamicInvoker class</a>.</p>
<hr width="100%">
</body></html>
```

Listing 7.2 – Web Service Invocation Sample Code

And finally, the way to invoke this service by first generating the stub interface and using this directly through WSIF's dynamic proxy, hiding all WSIF specifics from the client code. The stub interface used is the service interface as defined by the JAX-RPC specification.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><html><head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Author" content="Nirmal Mukhi">
<meta http-equiv="Content-Style-Type" content="text/css">
<title>Web Services Invocation Framework: Samples</title>
<link rel="stylesheet" href="wsif.css" type="text/css"></head>

<body alink="#0000ff" bgcolor="#ffffff" leftmargin="2" topmargin="2" marginwidth="2" marginheight="2">

<h2>
Web Services Invocation Framework:<br>
Invoking the SimpleSOAP Sample through a high level stub interface</h2>
First, you must <a href="../../../../doc/samples.html">set up the CLASSPATH in your environment</a>.
<p>This directory contains a file called <tt>Run.java</tt> that contains the <tt>main</tt> method. This
is the logic that uses the generated stub interface to run the sample. So you can run this class,
specifying on the command line the location of the WSDL file for the sample and the symbol for the
company whose stock quote you are interested in. For example, <br>
<tt>java simplesoap.client.stub.Run samples/simplesoap/StockquoteSOAP.wsdl IBM</tt></p>
<p>To generate the stub interface, you can use any tool that generates Java interfaces for WSDL
services using their port type descriptions, such as WSDL2Java from Axis. WSIF assumes a correspondence
between the generated Java interface and the WSDL port type that has its abstract description as
specified in the JAX-RPC specification. This particular sample used WSDL2Java in the following way:<br>
<tt>java org.apache.axis.wsdl.WSDL2Java ../../StockquoteSOAP.wsdl</tt><br>
After the tool finished running, we deleted all the generated files except
<tt>NetXmethodsServicesStockquoteStockQuotePortType.java</tt> (this is the java interface corresponding
to the port type and is all that is required by WSIF).</p>
<hr width="100%">
</body></html>
```

Listing 7.3 – Stub Interface Sample File

WSIF has a number of unit tests located in the tests directory. WSIF unit tests are loosely themed and test function in multiple providers.

For instance jms.JmsTest tests jms:address, jms:property and jms:propertyValue tags across the SoapJms, AxisJms and NativeJms providers. Users can either run a specific unit test or run all of them by running util.WSIFTestRunner. There are various listeners needed to run the unit tests (JMS2HTTPBridge, JMSAsyncListener and NativeJMSRequestListener) and unit tests automatically

start and stop the listeners that they need. All the tests need to find WSDL and other files on the local file system.

For SOAP and Axis over HTTP, all files called DeploymentDescriptor.xml in the samples and tests directories must be deployed to the web server. The web server must be running on the localhost web site using port 8080. The level of SOAP or Axis used on the server does not have to be the same as the level used by WSIF.

7.3 iTKo Lisa

The author has collected the required information from the product web site, following the indications related to the Web service Test Case given in the tutorial web pages[39].

The first step after the product installation consists of loading the web service test case and configuring the required attributes. For this example the tutorial proposes the “Execution of Web Services via SOAP” option.

Figure 7.4 displays the configuration page where the user has filled-in the information for the test case. The user must specify the URL where the WSDL file can be located and the URL where the web services to be tested has been published.

To correctly access the service the user must also supply the authentication data for the user that will access the service.

[39] http://www.itko.com/site/ws-testing/knowledge.jsp

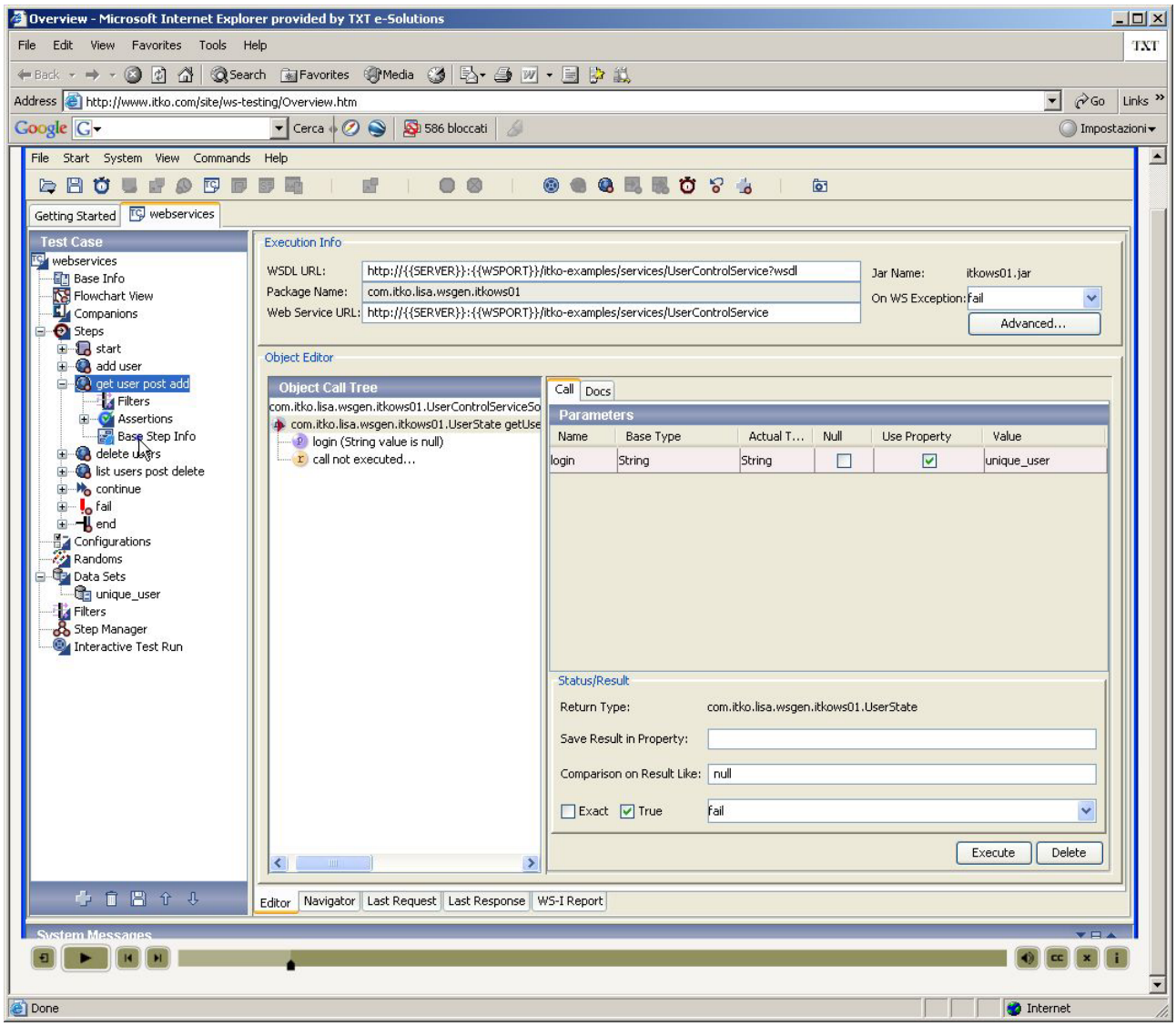

Figure 7.4: Configuration of a Webservice Sampler

The following Figure 7.5 shows the final step of the “Add User” menu tree.

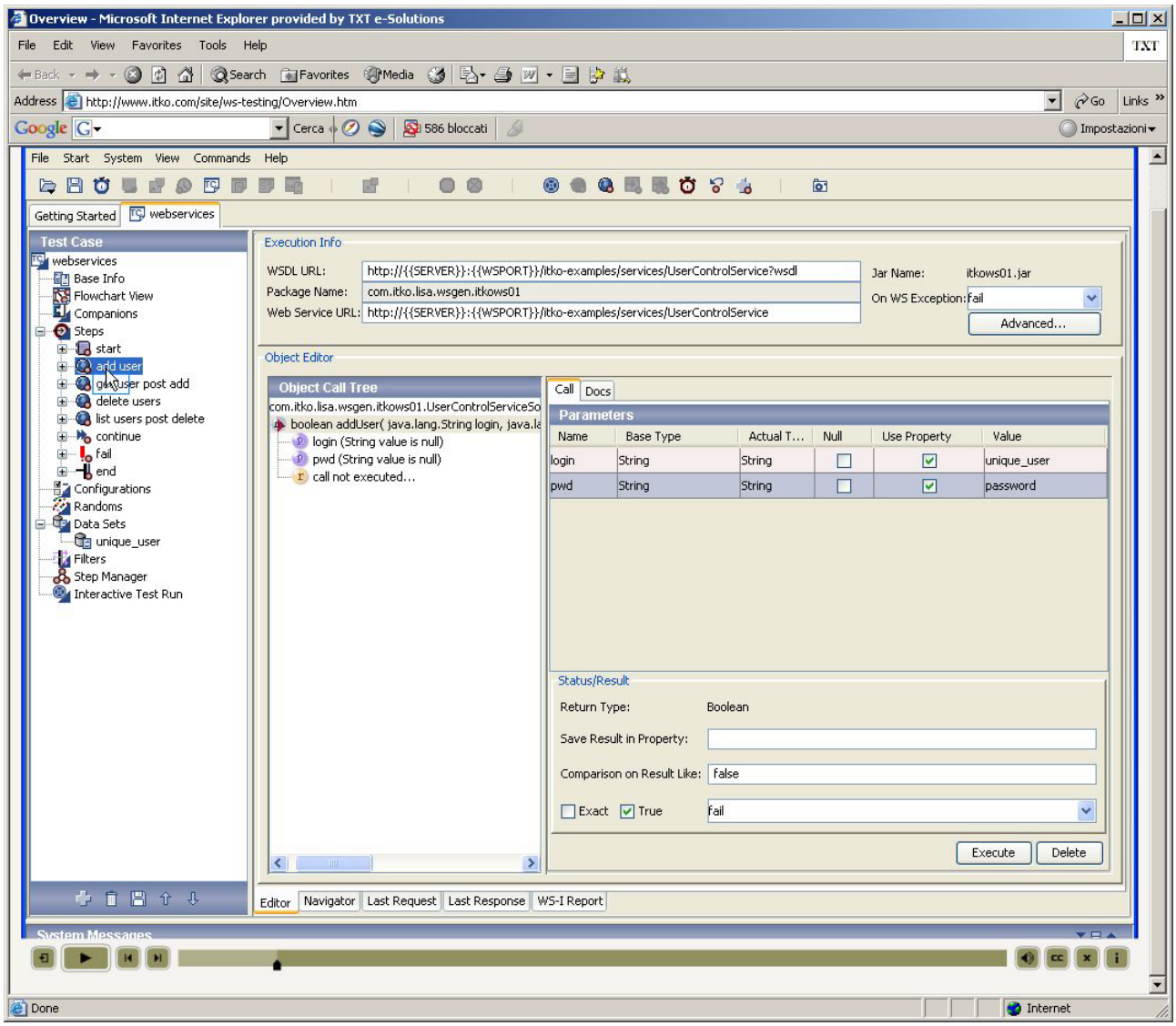

Figure 7.5 – Definition of User Data

To make testing sound and comprehensive it is possible to select from a predefined set of assertions that the application will verify against the response of the web service (see Figure 7.6).

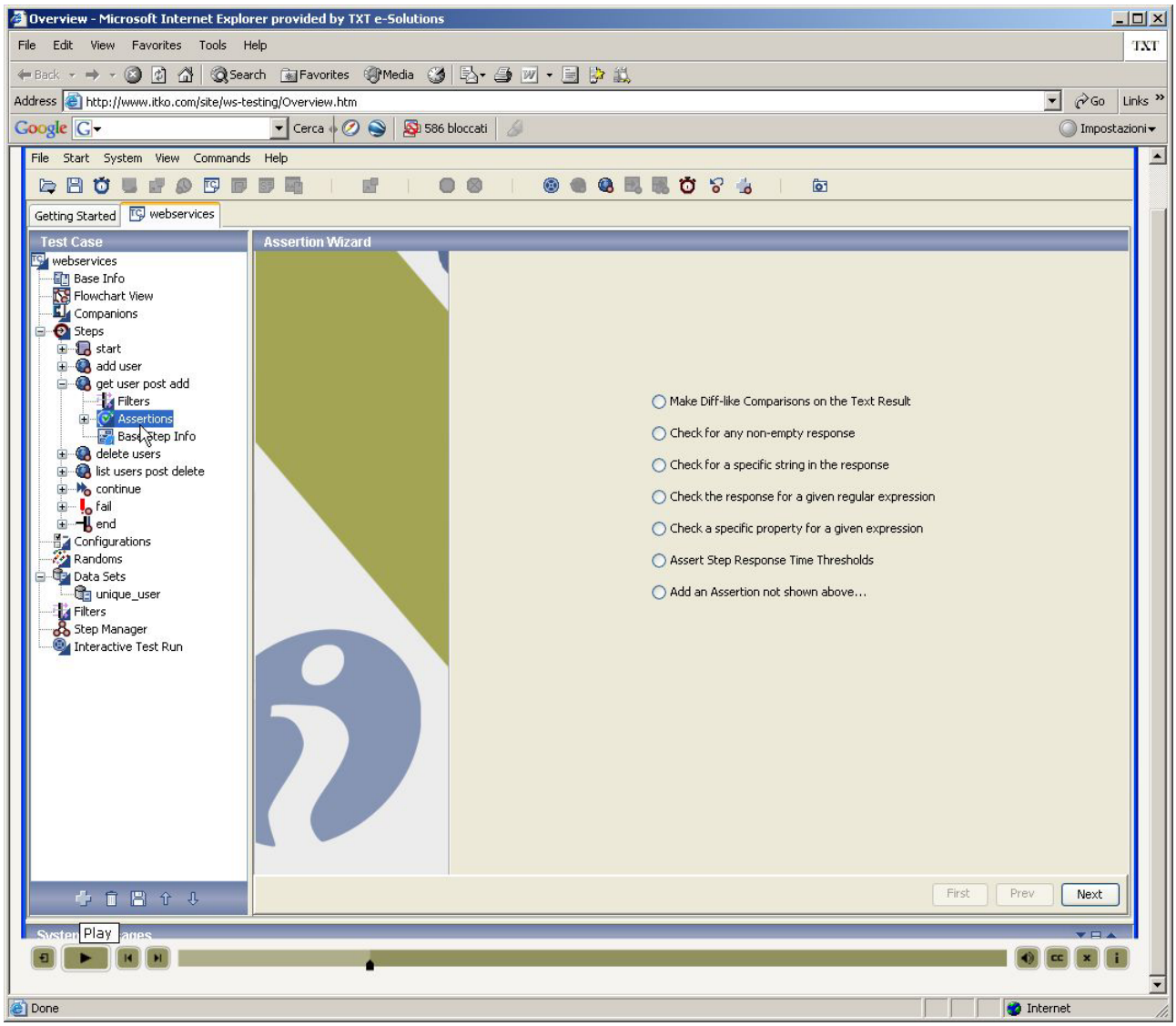

Figure 7.6: Specification of Assertions

In order to complete the service configuration the user must select the “Configurations” entry in the left-side tree view and finally start the test execution; this step can be accomplished either trough the Interactive Test Run in the tree view control or selecting the “Commands” entry in the main menu, like displayed in Figure 7.7.

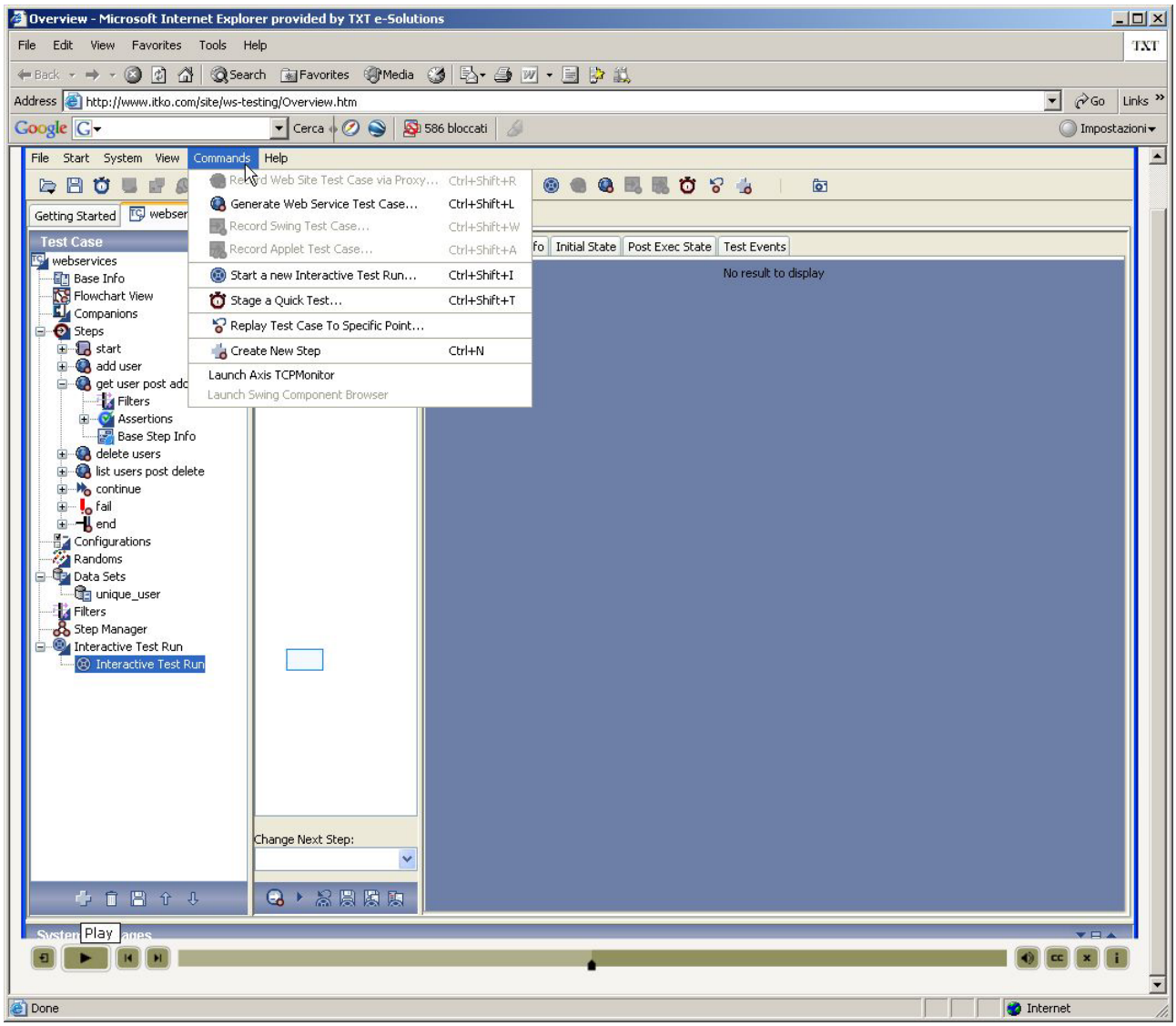

Figure 7.7: Execution of the Test

It is also possible to execute the test in background, waiting for completion, The user can thereafter review the results represented by responses returned after the web service invocation.

7.4 Parasoft SOATest

One of the greatest capabilities of SOAtest with regards to test suites is the ability to extract values from service responses via XPATH and then use the result as inputs in subsequent service calls. In this way, a typical "session" from a client making several service calls in a row can be elegantly represented in the tool. Test suites can contain many different types of tests; however, for the .NET developer, SOAP client tests will be the most used test type. Responses can be validated, as well, and SOAtest provides validation wizards to validate individual parameters (via static comparisons or regular expression matching), the entire message, or the structure of the response.

Above functional testing, all tests can be executed in a load testing scenario: user can select the tests and/or scenarios to execute as part of the load test and then select the performance counters to capture as part of the test execution. During execution, SOAtest captures the request/response metrics from the load test, as well as any performance metrics attached to the execution, and present the results in a unified view. SOAtest can simulate 100 virtual users on a single client.

7.4.1 WCF and SOAtest

The application works smoothly with ASMX Web services, while it is not so flawlessly when testing WCF services (Microsoft .NET framework). Basic HTTP binding (SOAP 1.2) worked as expected, but services exposed through wsHttpBinding did not. Even when using several permutations trying to get SOAtest to successfully send requests to WCF services exposed via wsHttpBinding, no avail. SOAtest was able to interpret the WSDL correctly. The application displays an interface to set parameters for the request, but the actual execution of the request fails; wsHttpBinding supports more WS* standards than SOAtest, and this is where the incompatibility lies. It is still possible to test WCF services using basicHttpBinding and a suitable workaround is to use multiple endpoints or to switch the binding when deploying.

7.4.2 Conclusion

SOAtest is an impressive tool for testing Web services in a variety of situations. The product exposes powerful functionalities and a really wide list of capabilities. However, because it is a Java-based application, it is not possible to test certain features. Over and above the "everyday" testing scenarios, SOAtest also allows for the establishment of policies that act as a kind of FxCop for WSDLs, penetration testing for security considerations, and even QoS and health reporting. Parasoft has committed to increasing SOAtest's .NET support, so I hope the issues of incompatibility with some WCF HTTP bindings will be resolved.

8. Conclusions

According to the evidence collected in the previous chapters and the results of this survey it seems that any software solution that will have to consider the INSPIRE requirements for web services and their interoperability cannot abstract from the "state of play" of a SOA architecture and its future evolution.

The problem with a "standard" SOA stack is that the well-defined and accepted "layers" associated with TCP/IP and OSI don't exist. There have been multiple attempts to define a SOA stack, but thus far there is no consensus regarding what layers should be included. While it's generally accepted that there's a transport (HTTP, FTP, SMTP, JMS) layer, a messaging layer (SOAP), a description layer (WSDL) and a discovery layer (UDDI, but increasingly vendors are demanding something more robust), there are a ton of other "layers" that potentially could be included: AAA, transaction management, coordination, and orchestration are just a few possibilities.

Outside the relatively safe "core" layers of the SOA stack, the ability to compose an architecture from different products and potentially from different vendors begins to fail. That's the implicit danger of an over-exploitation of a SOA stack.

It's a relatively safe assumption that no single vendor's "stack" will be adopted, and that's fine. As long as competing stacks are interoperable, that's a good thing. It would also be dangerous to start defining a SOA stack that goes behind the core layers necessary to provide connectivity and interoperable integration. Doing so limits the flexibility and interoperability of a SOA, and that's exactly the opposite of what a SOA is intended to provide.

While some vendors, most notably IBM, say they would like to see everyone agree on a single Web services stack – the protocols used to define, locate, implement and make Web services interact – it does not appear likely to happen.

Still there are still good reasons to think that open source stacks like Apache Axis 2.0, which is now part of IBM WebSphere, can become a reference point, anyway it is reasonable to expect that not all the vendors will settle on one standard stack.

Even if we grant that there should be (and is) a core SOA stack comprised of the transport, messaging, description, and delivery layers, still there is no reason to expect or even want a "standard" stack from a single vendor. The definition of a SOA is fluid and unique to each organization, so there's really no way to define a standard stack above and beyond the core layers for SOA that meets everyone's needs and goals.

Concerning interoperability and the UDDI standard supported by different producers, it seems that IBM (one of the creators) now believes that "SOAs have stretched UDDI web services standard to the limit, and that it's time for a new standard"[40]. Specifically, they say that UDDI does not support role-based access control to service information, lifecycle management, and comprehensive search. Therefore the industry needs a new registry standard. In the meantime, IBM will be offering a proprietary solution.

"The soon-to-be released WSRR v6.0.2 [WebSphere Service Registry and Repository] will include a UDDI synchronization framework, which will enable reasonable coexistence between WSRR and a separate UDDI registry. IBM even includes a separate UDDI registry with the product. But I don't think this makes up for the fact that WSRR does not directly support the UDDI protocol or for the fact that IBM has chosen to implement support for WSRR's proprietary registry protocol rather than UDDI in its plethora of runtime products. This is clearly a proprietary platform strategy."

IBM, however, says that UDDI was originally designed for Web services, which invoke point-to-point connections across the network. (In fact, it was designed to be the "Yellow Pages" of the e-business world.) But what enterprises need now is a registry standard that addresses the building-block, enterprise approach of SOA.

[40] Excerpt from a report in ITWeek. See http://blogs.zdnet.com/service-oriented/?p=864.

These statements make us think that there may soon be some kind of change in the scenario of standards used by applications for web service development and testing.

SOAs require different information about services than do Web services; UDDI "will not allow for role-based access to services, does not let companies manage a service's life cycle to enable governance, and does not allow for services to be searched".

A last comment on the article from Online News published in DMReview.com.[41].

This article reports that webMethods, Inc., a business integration and optimization software company, announced successful demonstration of interoperability of the WS-Policy Candidate Recommendation specifications using the UDDI (universal description, discovery and integration) standard. The first-ever interoperability evaluation event for the World Wide Web Consortium's (W3C) Web Services Policy 1.5 Framework and Attachment (WS-Policy) Candidate Recommendation specifications was hosted by webMethods.

During this recently concluded event, leading products that support the UDDI registry specification for lifecycle governance – the HP SOA Systinet Standard Edition and the webMethods Infravio X-Registry – were used to demonstrate interoperability with the Layer 7 SecureSpan XML Appliance, a widely deployed policy enforcement point, using the WS-Policy specifications. The development of a standards-based model for governance interoperability between policy enforcement points (PEP) and each policy's system-of-record represents a significant step towards enabling standards-based federated policy management for the enterprise.

One of the fundamental challenges that service-oriented architecture (SOA) works to address is orchestrating a series of very dynamic interactions between disparate Web services in accordance with enterprise-class standards for quality of service. WS-Policy helps to achieve this objective by facilitating agreement between producers and consumers of Web services. More specifically, WS-Policy provides a means for describing and communicating the capabilities and requirements of specific Web services in a coherent and reliable manner, ensuring that specific preconditions are fully met within each interaction.

As an underlying component of WS-Policy, the Web Services Policy 1.5 – Attachment specification can be used to bind specific policies to unique services via either WSDL (web services definition language) or UDDI. In the case of a UDDI registry, it defines how policies can be stored and accessed within an associated repository to deliver optimal performance. Successful testing paves the way for UDDI to be included, along with WSDL, as an acceptable means for policy exchange in the candidate recommendation section of the specification.

[41] Interoperability Demonstration of WS-Policy Using UDDI Policy Attachment.

Annex A: Acronyms

Acronyms for organisations / standards

BPMI.org	Business Process Management Initiative
INSPIRE	Infrastructure for Spatial InfoRmation in Europe
IETF	Internet Engineering Task Force
ISO	International Standards Organization
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium, also referred to as OpenGIS®
OSI	Open Systems Interconnection (ISO/OSI)
W3C	World Wide Web Consortium
WS-I	Web Services Interoperability Organization

Acronyms for technologies

AAA	Authenticate, Authorise and Audit
BPEL4WS	Business Process Execution Language for Web Services
BPML	Business Process Modelling Language
BPSS	Business Process Schema Specification
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Object Component Model
ebXML	Electronic Business using eXtensible Markup Language
EJB	Enterprise Java Beans
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
MTOM	Message Transmission Optimization Mechanism
ORB	Object Request Brokers
QoS	Quality of Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery, and Integration
WCF	Windows Communication Foundation
WSDL	Web Services Description Language
WS	Web Service
WSCDL	Web Service Choreography Definition Language
WSCI	Web Service Choreography Interface
WSIF	Web Services Invocation Framework
XML	eXtensible Markup Language

Annex B: References

The following table lists the sources referenced in the present document.

	Title	Author	Notes
[0]	Metadata State of Play	TXT e-Solutions	
[1]	INSPIRE Work Programme Transposition Phase 2007-2009	European Commission	http://www.ec-gis.org/inspire/reports/transposition/INSPIRE_IR_WP2007_2009_en.pdf
[2]	Detailed definitions on the INSPIRE Network Services	INSPIRE	http://www.ec-gis.org/inspire/reports/dt/ir_dev_process_network_services.pdf
[3]	Basic Profile Version 1.0	WS-I	http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html
[4]	ebXML Test Framework Committee Specification 1.0	OASIS	http://xml.coverpages.org/IIC-ebXMLTestFrame10.pdf
[5]	ebXML Test Framework v1.0[42]	OASIS	http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebxml-iic
[6]	SOA definition	Barry & Associates, Inc.	http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html
[7]	Business Process Execution Language for Web Services 1.1	BEA, IBM, Microsoft, SAP, Siebel	http://www.ibm.com/developerworks/library/specification/ws-bpel/
[8]	Understanding WSDL in a UDDI registry, Part 1	IBM	http://www.ibm.com/developerworks/webservices/library/ws-wsdl
[9]	Using WSDL in a UDDI Registry, Version 1.08	OASIS	http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm
[10]	Web Services Description Language (WSDL) 1.1	W3C	http://www.w3.org/TR/wsdl
[11]	WS-I Profile Test Assertion Document	WS-I	http://www.ws-i.org/Testing/Tools/2003/07/BasicProfileTestAssertions.xml
[12]	Sviluppare Web Services con Apache Axis	Gino L. Tesei	http://www.jugmilano.it/vqwiki/jsp/Wiki?action=action_view_attachment&attachment=SviluppareWebServicesConApacheAxis.ppt

[42] The final version is available at the following link: http://www.oasis-open.org/committees/download.php/10896/IIC_ebXMLTestFramework_v1.1_10_11_04_final_draft.zip

European Commission

EUR 23049 EN – Joint Research Centre – Institute for Environment and Sustainability
Title: Services State of Play, Compliance Testing and Interoperability Checking
Author(s): S. Viganó, M. Millot
Luxembourg: Office for Official Publications of the European Communities
2007 – 85 pp. – 21 x 29.7 cm
EUR – Scientific and Technical Research series – ISSN 1018-5593
ISBN 978-92-79-07820-0
DOI 10.2788/55814

Abstract

The document contains an inventory of existing solutions for compliance testing and interoperability checking of services, the assumption being that the services are web services. Even if the emphasis is on geographical information and therefore on Geographical Information Systems, the document describes applicable solutions outside the geographical Information System domain.

How to obtain EU publications

Our priced publications are available from EU Bookshop (http://bookshop.europa.eu), where you can place an order with the sales agent of your choice.

The Publications Office has a worldwide network of sales agents. You can obtain their contact details by sending a fax to (352) 29 29-42758.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

JRC
EUROPEAN COMMISSION

LB-NA-23049-EN-C

ISBN 978-92-79-07820-0